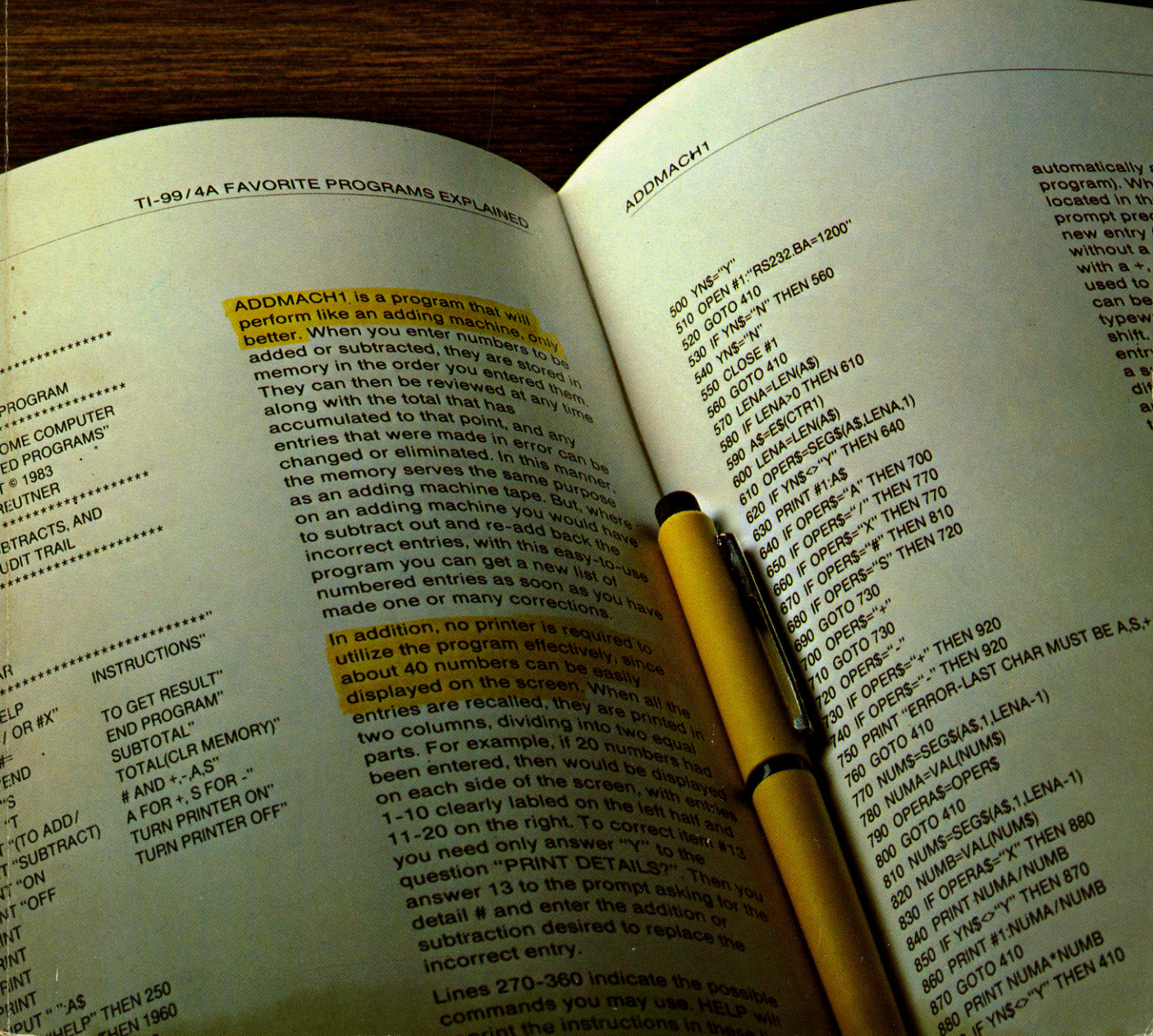
TI-99/4A FAVORITE PROGRAMS EXPLAINED



ADDMACH1 is a program that will perform like an adding machine, only better. When you enter numbers to be added or subtracted, they are stored in memory in the order you entered them. They can then be reviewed at any time along with the total that has accumulated to that point, and any entries that were made in error can be changed or eliminated. In this manner, the memory serves the same purpose as an adding machine tape. But, where on an adding machine you would have to subtract out and re-add back the incorrect entries, with this easy-to-use program you can get a new list of numbered entries as soon as you have made one or many corrections.

In addition, no printer is required to utilize the program effectively, since about 40 numbers can be easily displayed on the screen. When all the entries are recalled, they are printed in two columns, dividing into two equal parts. For example, if 20 numbers had been entered, then would be displayed on each side of the screen, with entries 1-10 clearly labled on the left half and 11-20 on the right. To correct item #13 you need only answer "Y" to the question "PRINT DETAILS?". Then you answer 13 to the prompt asking for the detail # and enter the addition or subtraction desired to replace the incorrect entry.

Lines 270-360 indicate the possible commands you may use. HELP will


```
500 YN$="Y"
510 OPEN #1:"RS232.BA=1200"
520 GOTO 410
530 IF YN$="N" THEN 560
540 YN$="N"
550 CLOSE #1
560 GOTO 410
570 LENA=LEN(A$)
580 IF LENA>0 THEN 610
590 A$=E$(CTR1)
600 LENA=LEN(A$)
610 OPER$=SEG$(A$,LENA,1)
620 IF YN$<>"Y" THEN 640
630 PRINT #1:A$
640 IF OPER$="A" THEN 700
650 IF OPER$="/" THEN 770
660 IF OPER$="X" THEN 770
670 IF OPER$="#" THEN 810
680 IF OPER$="S" THEN 720
690 GOTO 730
700 OPER$="+"
710 GOTO 730
720 OPER$="-"
730 IF OPER$="+" THEN 920
740 IF OPER$="-" THEN 920
750 PRINT "ERROR-LAST CHAR MUST BE A,S,+
760 GOTO 410
770 NUM$=SEG$(A$,1,LENA-1)
780 NUMA=VAL(NUM$)
790 OPERA$=OPER$
800 GOTO 410
810 NUM$=SEG$(A$,1,LENA-1)
820 NUMB=VAL(NUM$)
830 IF OPERA$="X" THEN 880
840 PRINT NUMA/NUMB
850 IF YN$<>"Y" THEN 870
860 PRINT #1:NUMA/NUMB
870 GOTO 410
880 PRINT NUMA*NUMB
890 IF YN$<>"Y" THEN 410
```

automatically
program). Wh
located in th
prompt pre
new entry
without a
with a +,
used to
can be
typew
shift,
entr
a s
di
a
t

TI-99/4A Favorite Programs Explained

Donald C. Kreutner

Que Corporation
Indianapolis

Library of Congress Catalog No.: LC 83-62061

ISBN 0-88022-050-3

About the Author

Donald C. Kreutner received his B.A. in history from Valparaiso University in 1967 and his M.S. in education from Indiana University in 1974. In addition to taking course work in computer science, business, and accounting, he has done work toward an M.B.A. degree at Indiana University.

Over the past fifteen years, Mr. Kreutner has worked as a teacher, linguist, data processing manager, programmer/analyst, and director of data processing.

Because of his familiarity with many programming languages and a wide variety of computers and microcomputers, Mr. Kreutner is an expert programmer. His teaching and linguistic experiences have made him an exceptional communicator, as well.

Editor
Virginia D. Noble, M.L.S.

Editorial Director
David F. Noble, Ph.D.

Managing Editor
Paul L. Mangin

Dedication

To my wife Janet,
my sons Derek and Jay,
and my parents Albert and Pauline

Table of Contents

Introduction

TI-99/4A Favorite Programs Explained is a unique book in several ways.

First, the book contains many interesting programs that are both practical and entertaining. Other books often contain program examples that are too undeveloped in the uses of BASIC or too short to be meaningful. *TI-99/4A Favorite Programs Explained* is designed to help users learn better the applications of TI BASIC. The book is fun to use as well.

Second, *TI-99/4A Favorite Programs Explained* provides with each program clear and concise explanations of what actually takes place within the program. On reading (and rereading) these discussions, you can learn more about why the programs work the way they do.

All the programs have been written intentionally in standard TI BASIC. Because many readers do not have the TI EXTENDED BASIC cartridge, the programs in this book can be run on the simplest TI-99/4A setup. With only the computer and perhaps a cassette recorder, practically every program can be run satisfactorily. But if you do have available a disk drive or a printer, many of the programs here can already make full use of these devices without any changes needed in the BASIC code.

Finally, this book is unique because it provides programs for one of the most powerful computers available for its price. If you already own a TI-99/4A, with its many built-in capabilities, you have made a good investment. You do not need to worry about your computer becoming quickly outdated, even as newer and different machines become available. Texas Instruments has greatly enhanced the value of its product by making it upwardly compatible with its latest generation of low-cost peripherals, the HEX-BUS™ interface products. These new and exciting devices, which will enable you to use your TI-99/4A as a real data-processing machine, are described fully in the Appendix.

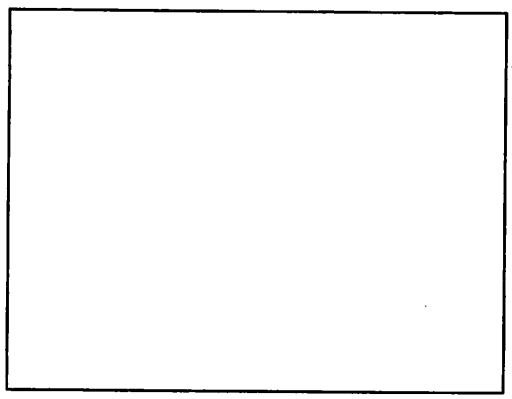
Despite the large amount of testing and proofreading that has been done in the preparation of this book, neither it nor the programs are necessarily perfect. Every program, however, has been thoroughly tested on the TI-99/4A. Many of the programs should be convertible to run on TI's new Compact Computer, the CC-40, with minimal changes. As you read *TI-99/4A Favorite Programs Explained*, you will acquire techniques in TI BASIC that may be helpful in other ways. After reading the book, you will have available useful programs for both work and play.

To key in any program, simply follow the guidelines in your TI reference manuals. If you need to modify any of the lines (for instance, to put your own DATA statements in the programs), you may need to skip the line numbers for the "test" DATA statements in the programs. Just key in your own statements, instead. (Be certain to end your DATA statements with the data element "END," or whatever ends the DATA statements in the book, so that the program can identify the end of data.)

Chapters 1, 2, and 3 contain different kinds of programs, including math and problem-solving, games and miscellaneous, and business and educational applications. Dividing the programs into such groupings was not an easy task since many of the programs are quite versatile. For instance, some of the math programs are also educational, and some miscellaneous programs have uses in business, too.

Whenever a printer is allowed within a program, an OPEN statement is necessary for the device, such as "RS232.BA=1200." This particular statement is needed for a system whose printer is attached to an RS-232 card and set at a baud rate of 1200 BPS (bits per second). If your printer has a different baud rate (or other options), you may need to alter these OPEN statements to suit the printer you are using.

In *TI-99/4A Favorite Programs Explained*, an illustration that is contained within a box, as shown below, represents a screen illustration:



Throughout the book these boxes show examples of pictures or displays on your TV or monitor that result from running the programs being described.

To run any program, key it in as printed and then save it to cassette or disk immediately (so that you do not have to retype the program again), as in the following:

```
>SAVE CS1
```

or

```
>SAVE DSK1.(program name)
```

You can reload any program by typing

```
>OLD CS1
```

or

```
>OLD DSK1.(program name)
```

Then you can run the program by simply stating

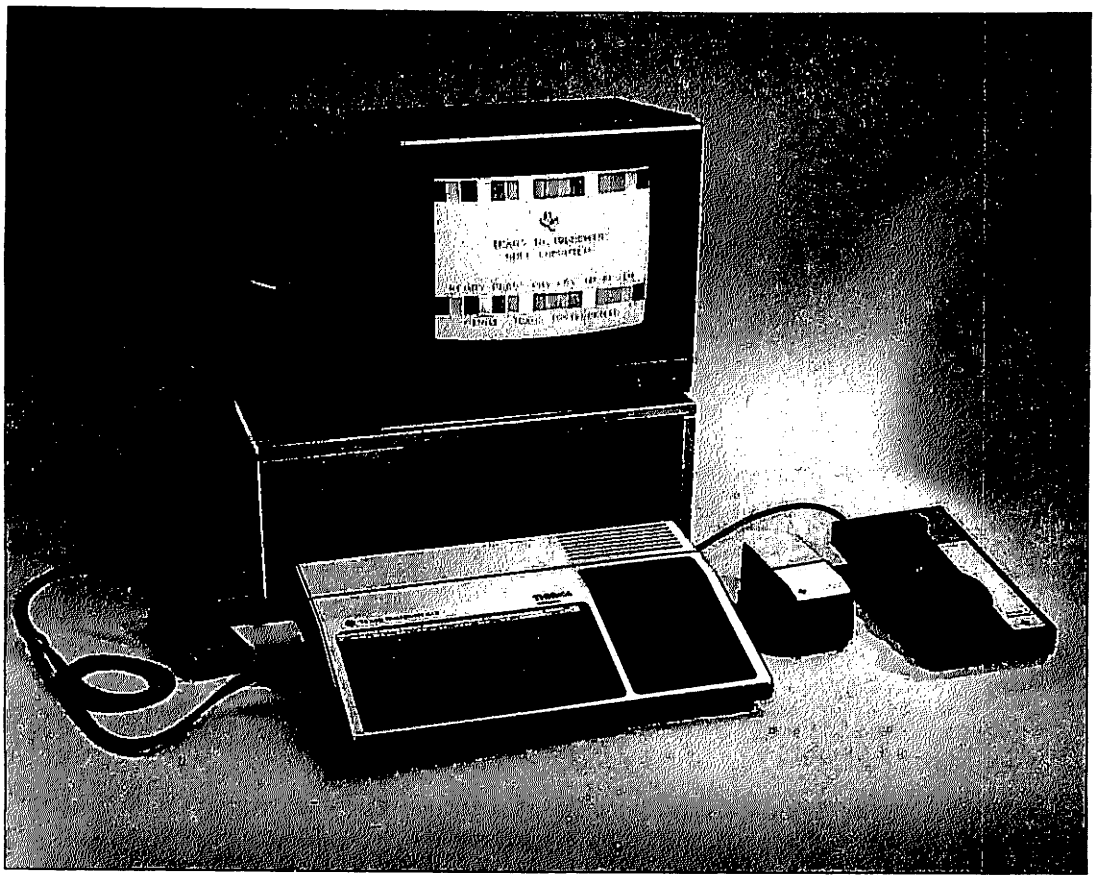
```
>RUN
```

In addition to the methods described in the book for terminating the programs, any program can be ended without damage by holding down the FCTN key while simultaneously depressing 4 (CLEAR).

Whenever a program asks you to key data by an INPUT statement, you must hit the ENTER key (sometimes called the RETURN key) after keying the desired input.

An acknowledgment should be made to Texas Instruments for providing the fine photographs that appear in this book and for permitting the use of portions of their TI manual and *TI-99/4A Reference Card* in Chapter 4.

Thanks also are given to Donna S. Padgett for permission to use her programs, TICTACTO and TICHUMAN.



The TI-99/4A Home Computer

1

Math and Problem-Solving Programs

ADDMACH1

```
100 REM * ************************
110 REM *          ADDMACH1        *
120 REM *                          *
130 REM *       COPYRIGHT 1983     *
140 REM *     DONALD C. KREUTNER   *
150 REM * ************************
160 REM ADDS, SUBTRACTS, AND
170 REM KEEPS AUDIT TRAIL
180 REM ************************
190 DIM E$(200)
200 CTR1=0
210 TOTAL=0
220 CALL CLEAR
230 PRINT "************************"
240 PRINT "HELP          INSTRUCTIONS"
250 PRINT "#/ OR #X"
260 PRINT "#=            TO GET RESULT"
270 PRINT "END           END PROGRAM"
280 PRINT "S             SUBTOTAL"
290 PRINT "T             TOTAL(CLR MEMORY)"
300 PRINT "(TO ADD/      # AND +,-,A,S"
310 PRINT "SUBTRACT)     A FOR +, S FOR -"
320 PRINT "ON            TURN PRINTER ON"
```

ADDMACH1 is a program that performs like an adding machine, only better. When you key numbers to be added or subtracted, they are stored in memory in the same order in which you enter them. They can be reviewed later at any time, along with the total that has accumulated to that point. Any entries made in error can be changed or eliminated. In this manner, the memory serves the same purpose as an adding machine tape. On an adding machine, however, you have to subtract out and add back incorrect entries. With ADDMACH1 you can get a new list of numbered entries as soon as you have made one or more corrections.

In addition, ADDMACH1 does not need a printer, since about 40 numbers can be easily displayed on the screen. When all the entries are recalled, they are displayed in two columns. For example, if 20 numbers have been entered, ten would be displayed on each side of the screen, with entries

```
330 PRINT "OFF          TURN PRINTER OFF"
340 PRINT
350 PRINT
360 PRINT
370 PRINT
380 INPUT " ":A$
390 IF A$="HELP" THEN 220
400 IF A$="END" THEN 1930
410 IF A$="S" THEN 980
420 IF A$="T" THEN 1000
430 IF A$="ON" THEN 460
440 IF A$="OFF" THEN 500
450 GOTO 540
460 IF YN$="Y" THEN 490
470 YN$="Y"
480 OPEN #1:"RS232.BA=1200"
490 GOTO 380
500 IF YN$="N" THEN 530
510 YN$="N"
520 CLOSE #1
530 GOTO 380
540 LENA=LEN(A$)
550 IF LENA>0 THEN 580
560 A$=E$(CTR1)
570 LENA=LEN(A$)
580 OPER$=SEG$(A$,LENA,1)
590 IF YN$<>"Y" THEN 610
600 PRINT #1:A$
610 IF OPER$="A" THEN 670
620 IF OPER$="/" THEN 740
630 IF OPER$="X" THEN 740
640 IF OPER$="=" THEN 780
650 IF OPER$="S" THEN 690
660 GOTO 700
670 OPER$="+"
680 GOTO 700
690 OPER$="-"
700 IF OPER$="+" THEN 890
710 IF OPER$="-" THEN 890
720 PRINT "ERROR - LAST CHAR MUST BE A,S,+,-"
```

1-10 clearly labeled on the left half and 11-20 on the right. To correct item #13, you need only answer **Y** (for Yes) to the question "PRINT DETAILS?" Then you answer **13** to the prompt asking for the detail # and enter the addition or subtraction desired to replace the incorrect entry.

Lines 240-330 indicate the possible commands you may use. **HELP** will redisplay the instructions in these lines. The instructions come up on the screen automatically at the startup of the program.

Whenever the cursor is located in the second position without a preceding prompt, you may make a new entry by keying a number with or without a decimal point and ending the number with a + or -, or an **A** or **S**. An **A** (for add) and an **S** (for subtract) can be used for quick entry of a + or - since the letters can be easily accessed without a shift on the TI typewriter-style keyboard. With this method, you can enter one entry after another until you want either a subtotal (enter an **S** alone) or a total (**T**). The difference between the two is that **T** will automatically clear the memory and total accumulations, whereas **S** will show the total to that point, allowing you to continue making entries after viewing the subtotal, as well as making any needed corrections.

The commands **ON** and **OFF** will turn a printer on or off to provide a printed copy of the entries you are making on the screen. You can turn the printer on or off whenever desired. As mentioned earlier, however, you can use ADDMACH1 without a printer and therefore save paper—either printer paper or adding machine tape.

```
730 GOTO 380
740 NUM$=SEG$(A$,1,LENA-1)
750 NUMA=VAL(NUM$)
760 OPERA$=OPER$
770 GOTO 380
780 NUM$=SEG$(A$,1,LENA-1)
790 NUMB=VAL(NUM$)
800 IF OPERA$="X" THEN 850
810 PRINT NUMA/NUMB
820 IF YN$<>"Y" THEN 840
830 PRINT #1:NUMA/NUMB
840 GOTO 380
850 PRINT NUMA*NUMB
860 IF YN$<>"Y" THEN 380
870 PRINT #1:NUMA*NUMB
880 GOTO 380
890 NUM$=SEG$(A$,1,LENA-1)
900 NUM1=VAL(NUM$)
910 IF OPER$="+" THEN 940
920 TOTAL=TOTAL-NUM1
930 GOTO 950
940 TOTAL=TOTAL+NUM1
950 CTR1=CTR1+1
960 E$(CTR1)=NUM$&OPER$
970 GOTO 380
980 PRINT "SUBTOTAL = ";TOTAL
990 GOTO 1010
1000 PRINT "TOTAL = ";TOTAL
1010 IF YN$<>"Y" THEN 1060
1020 IF A$="S" THEN 1050
1030 PRINT #1:"TOTAL = ";TOTAL
1040 GOTO 1060
1050 PRINT #1:"SUBTOTAL = ";TOTAL
1060 INPUT "PRINT DETAILS? ":PYN$
1070 IF A$="S" THEN 1090
1080 IF A$="T" THEN 1110
1090 IF PYN$<>"Y" THEN 380
1100 GOTO 1120
1110 IF PYN$<>"Y" THEN 1900
1120 PRINT
```

One final feature is that you may obtain intermediate results which do not interfere with memory storage. These are accomplished by entering a number followed by a **/** (for division) or an **X** (for multiplication) and by pressing **ENTER**. Then by entering the number to be multiplied by or divided by, followed by an equal sign, and by pressing **ENTER**, you will obtain the answer. For example, if you first key **12.34X** and an **ENTER**, then **25=** and an **ENTER**, you will get the answer 308.5. Then you can add or subtract 308.5 by entering **308.5+** or **308.5-** and by pressing **ENTER**.

You can add or subtract the same number repetitively without re-entering it. Just enter one time the number to be added or subtracted and key **ENTER** for each calculation. For instance, by entering **25.37-** and keying **ENTER ENTER ENTER ENTER ENTER ENTER**, you are subtracting 25.37 six times. Lines 460-530 open and close the printer (if one is used) as you enter **ON** or **OFF**. When the printer has been turned on, a flag is set to "Y" (YN$ in line 470) or "N" (line 510). The program, then, is always aware of whether to print the desired details to the printer as well as display them on the screen.

Lines 610-720 evaluate the arithmetic operator keyed as the last character of an entry. If the character is not an **A**, **/**, **X**, **=**, **S**, **+**, or **-**, the operator is considered invalid, and an error message is displayed (line 720).

Lines 740-770 evaluate the number keyed before an **X** or a **/** so that the next number keyed before an equal sign can be evaluated in lines 780-880. In lines 810 and 830,

```
1130 CTR2=0
1140 CTR1A=CTR1/2
1150 IF CTR1A=INT(CTR1A)THEN 1170
1160 CTR1A=INT(CTR1A)+1
1170 FOR I=1 TO CTR1A
1180 I$=STR$(I)
1190 LENI=LEN(I$)
1200 IF LENI<2 THEN 1230
1210 I2$=I$&". "
1220 GOTO 1240
1230 I2$=" "&I$&". "
1240 PRINT I2$;E$(I);
1250 IF YN$<>"Y" THEN 1270
1260 PRINT #1:I2$;E$(I);
1270 IX=I+CTR1A
1280 IF IX>CTR1 THEN 1380
1290 IX$=STR$(IX)
1300 LENIX=LEN(IX$)
1310 IF LENIX<2 THEN 1340
1320 IX2$=IX$&". "
1330 GOTO 1350
1340 IX2$=" "&IX$&". "
1350 PRINT TAB(15);IX2$;E$(IX);
1360 IF YN$<>"Y" THEN 1380
1370 PRINT #1:TAB(15);IX2$;E$(IX);
1380 PRINT
1390 IF YN$<>"Y" THEN 1410
1400 PRINT #1
1410 NEXT I
1420 IF YN$<>"Y" THEN 1490
1430 PRINT #1
1440 IF A$="S" THEN 1470
1450 PRINT #1:"TOTAL = ";TOTAL
1460 GOTO 1480
1470 PRINT #1:"SUBTOTAL = ";TOTAL
1480 PRINT #1:"**************************"
1490 PRINT
1500 IF A$="S" THEN 1530
1510 PRINT "TOTAL = ";TOTAL
1520 GOTO 1540
```

the intermediate division result is listed on the screen and/or to the printer. Likewise, an intermediate multiplication product is printed in lines 850 and 870.

Lines 890-970 accumulate the number keyed into the total, either adding or subtracting, depending on the ending operator (+, -, **A**, or **S**). Furthermore, the entry is saved in the memory array E$ (line 960). Note that if an **S** or **A** was keyed, a - or + replaces the operator keyed in memory.

The subtotal or total is printed in lines 980-1550. The number of entries contained in memory (CTR1 in line 1140) is divided by two to determine how many rows of two columns should be displayed or printed.

As previously mentioned, each item is clearly numbered so that any entry can be easily located and changed. Lines 1560-1890 allow you to make changes or deletions to the displayed entries. To replace an entry with a correction, you need only to key the new entry after responding to the prompt that asks for the detail number in line 1580. The old entry is then either subtracted from the total (if the entry had been added to it) or added to the total (if the entry had been subtracted from it). The keyed replacement is stored in the same location of the memory array.

Lines 1900-1920 clear the TOTAL accumulator and set the pointer for the memory array back to zero.

```
1530 PRINT "SUBTOTAL = ";TOTAL
1540 PRINT "**************************"
1550 IF A$="T" THEN 1900
1560 INPUT "CHG DETAIL? ":CYN$
1570 IF CYN$<>"Y" THEN 380
1580 INPUT "DETAIL#: ":DN
1590 IF DN>CTR1 THEN 1560
1600 INPUT "NEW VALUE: ":A$
1610 LENA=LEN(A$)
1620 IF LENA=0 THEN 1600
1630 OPER$=SEG$(A$,LENA,1)
1640 IF OPER$="A" THEN 1670
1650 IF OPER$="S" THEN 1690
1660 GOTO 1700
1670 OPER$="+"
1680 GOTO 1700
1690 OPER$="-"
1700 IF OPER$="+" THEN 1740
1710 IF OPER$="-" THEN 1740
1720 PRINT "ERROR - LAST CHAR MUST BE A,
     S,+,-"
1730 GOTO 1600
1740 NUM$=SEG$(A$,1,LENA-1)
1750 NUM1=VAL(NUM$)
1760 LENE=LEN(E$(DN))
1770 OPERE$=SEG$(E$(DN),LENE,1)
1780 NUME$=SEG$(E$(DN),1,LENE-1)
1790 NUME=VAL(NUME$)
1800 IF OPERE$="+" THEN 1830
1810 TOTAL=TOTAL+NUME
1820 GOTO 1840
1830 TOTAL=TOTAL-NUME
1840 IF OPER$="+" THEN 1870
1850 TOTAL=TOTAL-NUM1
1860 GOTO 1880
1870 TOTAL=TOTAL+NUM1
1880 E$(DN)=NUM$&OPER$
1890 GOTO 1560
1900 CTR1=0
```

(Continued in next column)

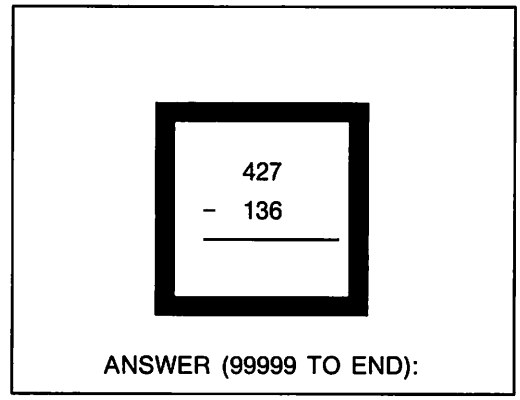
(Continued from previous column)

```
1910 TOTAL=0
1920 GOTO 380
1930 IF YN$<>"Y" THEN 1950
1940 CLOSE #1
1950 END
```

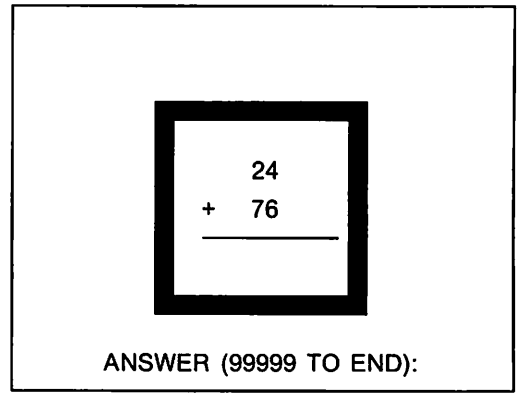
```
SUBTOTAL = 987.79
 1. 24.00+     18. 54.00-
 2. 24.00+     19. 35.00+
 3. 24.00+     20. 35.00+
 4. 24.00+     21. 35.00+
 5. 24.00+     22. 35.00+
 6. 24.00+     23. 35.00+
 7. 24.00+     24. 35.00+
 8. 24.00+     25. 41.21-
 9. 34.00+     26. 65.00+
10. 34.00+     27. 65.00+
11. 34.00+     28. 65.00+
12. 34.00+     29. 65.00+
13. 34.00+     30. 23.00+
14. 34.00+     31. 23.00+
15. 34.00+     32. 23.00+
16. 34.00+     33. 23.00+
17. 34.00+     34. 23.00+

SUBTOTAL =  987.79
**************************
```

ADDSUB1

```
100 REM ****************************
110 REM *                          *
120 REM *          ADDSUB1         *
130 REM *       COPYRIGHT 1983     *
140 REM *                          *
150 REM *     DONALD C. KREUTNER   *
160 REM ****************************
170 REM ADDSUB1
180 RANDOMIZE
190 NEG$="N"
200 CALL CLEAR
210 INPUT "WITH BORDER (Y/N): ":YN$
220 INPUT "TOP # FROM 1 TO (2/999): ":T1
230 IF T1<2 THEN 220
240 IF T1>999 THEN 220
250 INPUT "BOTTOM # 1 TO (2/999): ":B1
260 IF B1<2 THEN 250
270 IF B1>999 THEN 250
280 ERRS=0
290 CORR=0
300 INPUT "ADD, SUB, OR BOTH (A/S/B): ":ASB$
310 IF ASB$="A" THEN 350
320 IF ASB$="S" THEN 350
330 IF ASB$="B" THEN 350
340 GOTO 300
350 B$="FFFFFFFFFFFFFFFF"
360 U$="000000FFFF000000"
370 CALL CHAR(96,B$)
380 CALL CHAR(97,U$)
390 CALL CLEAR
400 IF YN$="N" THEN 450
410 CALL HCHAR(8,10,96,12)
420 CALL HCHAR(18,10,96,12)
430 CALL VCHAR(9,10,96,9)
440 CALL VCHAR(9,21,96,9)
450 NUM1=INT(T1*RND)+1
460 NUM2=INT(B1*RND)+1
```



ADDSUB1 is a mathematical skills builder designed for improving performance in addition and subtraction.



As is evident in the illustrations, the program will draw a border in the middle of the screen (if desired) in line 210. By answering any number from **2** to **999** in lines 220 and 250, the user can easily select the upper and lower ranges of the top and bottom

```
470 IF NEG$<>"N" THEN 520
480 IF NUM1>NUM2 THEN 520
490 NUM1X=NUM1
500 NUM1=NUM2
510 NUM2=NUM1X
520 IF ASB$="S" THEN 580
530 IF ASB$="A" THEN 560
540 OP1=INT(2*RND)+1
550 GOTO 590
560 OP1=1
570 GOTO 590
580 OP1=2
590 IF OP1=1 THEN 600
600 NUM1$=STR$(NUM1)
610 NUM2$=STR$(NUM2)
620 IF OP1=1 THEN 650
630 OP$="-"
640 GOTO 660
650 OP$="+"
660 LEN1=LEN(NUM1$)
670 LEN2=LEN(NUM2$)
680 J=18
690 FOR I=LEN1 TO 1 STEP -1
700 J=J-1
710 V=ASC(SEG$(NUM1$,I,1))
720 CALL VCHAR(12,J,V,1)
730 NEXT I
740 J=18
750 FOR I=LEN2 TO 1 STEP -1
760 J=J-1
770 V=ASC(SEG$(NUM2$,I,1))
780 CALL VCHAR(13,J,V,1)
790 NEXT I
800 OP2=ASC(OP$)
810 CALL VCHAR(13,14,OP2,1)
820 CALL HCHAR(14,14,97,4)
830 INPUT "ANSWER (99999 TO END): ":X
840 IF X=99999 THEN 1000
850 IF OP1=2 THEN 880
860 IF X=NUM1+NUM2 THEN 960
```

numbers to be added or subtracted. The totals of the correct and incorrect answers are set to zero in lines 280 and 290. You can select addition only, subtraction only, or a mixture of both, selected randomly. Notice that the **RANDOMIZE** statement in line 180 causes a different set of random selections each time a program with **RND** functions is run.

Two special characters are defined for this program: a solid block and a horizontal line through the middle of the character grid.

All F's in the string B$ cause the solid block to be defined by the **CHAR** statement in line 370, while the 0's in U$ cause empty horizontal bands above and below the two solid bands. The solid square is defined as character code ASCII 96 in line 370, and the underline is defined as character code ASCII 97 in line 380. The square is used to draw the rectangular box in which each generated problem is displayed. The underline character is used to draw the dark bar beneath the problem, just as one would draw it on paper. The random selection process for the top and bottom numbers is given in lines 450 and 460. If you want a random number between 1 and 50, for example, use the statement

```
10 NUM1=INT(50*RND)+1
```

To avoid negative answers, the value N is given to NEG$ in line 190.

To display the problem in the middle of the screen, the two numbers are converted to character strings in NUM1$ and NUM2$. The lengths of these two alphanumeric (another word for string) items are deter-

```
870 GOTO 890
880 IF X=NUM1-NUM2 THEN 960
890 CALL CLEAR
900 IF OP1=2 THEN 930
910 PRINT "INCORRECT";NUM1;"+";NUM2;"=";
    NUM1+NUM2
920 GOTO 940
930 PRINT "INCORRECT";NUM1;"-";NUM2;"=";
    NUM1-NUM2
940 ERRS=ERRS+1
950 GOTO 980
960 CORR=CORR+1
970 GOTO 390
980 INPUT " ":X$
990 GOTO 390
1000 CALL CLEAR
1010 PRINT "**********************"
1020 PRINT "*";TAB(23);"*"
1030 PRINT "*CORRECT =";CORR;TAB(23);"*"
1040 PRINT "*";TAB(23);"*"
1050 PRINT "*ERRORS=";ERRS;TAB(23);"*"
1060 PRINT "*";TAB(23);"*"
1070 IF CORR>0 THEN 1100
1080 IF ERRS>0 THEN 1100
1090 ERRS=1
1100 PRINT "*SCORE =";INT(CORR/(CORR+ERRS)
     *100);"%";TAB(23);"*"
1110 PRINT "*";TAB(23);"*"
1120 PRINT "**********************"
1130 PRINT
1140 PRINT
1150 END
```

mined in lines 660 and 670. The characters of the top and bottom numbers are then moved in, one digit at a time, from the 17th position on the screen, right to left. This move is accomplished by two **FOR-NEXT** loops that have a negative step (that is, I decreases from the length of the string to 1). At the same time, each digit is moved to the 17th position of the screen, then the 16th, etc., in rows 12 and 13, using the **VCHAR** calls in lines 720 and 780.

Note that when only one character at a time is being moved to the screen, **VCHAR** or **HCHAR** will work equally well. But if a character is being repeated more than once (as in lines 410-440 to draw the rectangle), then **VCHAR** repeats the character vertically and **HCHAR**, horizontally. With TI EXTENDED BASIC, instead of moving characters one at a time, the program can display a whole string, starting at a certain row and column. Obviously, this method is much faster!

After the problem is displayed, you are prompted at the bottom for an answer. If the response you give is correct, the program goes on to another problem. If your answer is incorrect, you are shown the correct answer. You must then press **ENTER** to continue.

To end the program, key **99999** as the answer to any problem. You are then shown a summary of your correct and incorrect responses.

BSKTSTAT

```
 10 REM * ************************ *
 20 REM *                          *
 30 REM *          BSKTSTAT        *
 40 REM *                          *
 50 REM *       COPYRIGHT 1983     *
 60 REM *     DONALD C. KREUTNER   *
 70 REM * ************************ *
100 REM BSKTSTAT
110 ECTR=0
120 DIM D$(300)
130 DIM P$(100)
140 GOSUB 3500
150 B$=""
160 CALL CLEAR
170 PRINT "ED   ENTER DETAILS"
180 PRINT "RD   READ DETAILS FILE"
190 PRINT "WD   WRITE DETAILS FILE"
200 PRINT "SD   SORT DETAILS"
210 PRINT "LD   LIST DETAILS"
220 PRINT "PD   PRINT DETAILS"
230 PRINT "C    CLEAR DETAILS"
240 PRINT "RP   READ PLAYER FILE"
250 PRINT "WP   WRITE PLAYER FILE"
260 PRINT "END  END PROGRAM"
270 PRINT "**************************"
280 PRINT
290 PRINT
300 PRINT
310 INPUT "ACTION: ":AC$
320 IF AC$="ED" THEN 880
330 IF AC$="RD" THEN 1660
340 IF AC$="WD" THEN 1850
350 IF AC$="SD" THEN 2000
360 IF AC$="LD" THEN 2170
370 IF AC$="PD" THEN 2290
380 IF AC$="END" THEN 3590
390 IF AC$="C" THEN 3470
```

BSKTSTAT is a multipurpose, basketball statistics accumulation program. As indicated by the menu in lines 170-260, this program will enter new details for a team's game statistics. Old details can be read from a file prior to entering new details in order to obtain cumulative totals. Details can then be sorted and listed as raw data, or the details and total statistics can be displayed on the screen or printed on the printer, or both. A player file can be read in from cassette or disk, or a new player file with names and numbers can be entered and written to tape or disk.

Based on the action requested in line 310, the allowable options are taken by the program (lines 310-420).

Note that when the program starts, the subroutine in lines 3500-3530 is executed, blanking out the names of the player string array P$. This array will hold up to 100 names that can then be accessed by player number. The **GOSUB** statement is particularly useful for accomplishing a routine that will be done at more than one place within a program. Here **GOSUB** allows you to perform a routine of four lines simply by stating **GOSUB** 3500. Control is then returned to the next sequential line (150) after the **RETURN** statement is encountered in line 3530.

Lines 440-610 accomplish the reading into P$ of a player name file that had been previously written either to disk or tape, using the **WP** (write player file) command of this program. Line 450 controls whether the

```
400 IF AC$="RP" THEN 440
410 IF AC$="WP" THEN 630
420 GOTO 160
430 REM ********************
440 GOSUB 3500
450 INPUT "CASSETTE/DISK (C/D): ":CD$
460 IF CD$<>"C" THEN 490
470 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
480 GOTO 510
490 INPUT "DSK1.FILENAME: ":FILE1$
500 OPEN #1:FILE1$,INPUT
510 INPUT #1:I$
520 IF I$="END" THEN 590
530 PN$=SEG$(I$,1,2)
540 LENI=LEN(I$)
550 PNAME$=SEG$(I$,3,LENI-2)
560 PN=VAL(PN$)
570 P$(PN)=PNAME$
580 GOTO 510
590 CLOSE #1
600 PRINT "END OF PLAYER READ"
610 GOTO 160
620 REM ********************
630 INPUT "CASSETTE/DISK (C/D): ":CD$
640 IF CD$<>"C" THEN 670
650 OPEN #1:"CS1",OUTPUT,INTERNAL,
    FIXED 128
660 GOTO 690
670 INPUT "DSK1.FILENAME: ":FILE1$
680 OPEN #1:FILE1$,OUTPUT
690 INPUT "PLAYER ##: ":PN
700 IF PN=999 THEN 830
710 PN$=STR$(PN)
720 IF PN>99 THEN 690
730 IF PN<0 THEN 690
740 IF PN>9 THEN 760
750 PN$="0"&PN$
760 INPUT "PLAYER NAME (20 CHARS): ":
    PNAME$
770 LENPN=LEN(PNAME$)
```

file will be read from a cassette recorder or a disk drive. The answer you key will then cause one or the other type of file to be opened for input (lines 470 and 500). Both files are sequential in nature, and I$ is input from the file. The first two characters are the player number. From position 3 to the end of the string is the player's name. Note that the **VAL** function of line 560 obtains the numeric value of the player number string.

Lines 630-860 similarly write a player file to tape or disk. Be sure to note, however, that you will have to enter the player number and name for every player you wish to have in the file. Once you have the file stored on disk or cassette, you can later recall that "team" whenever you want to access the same information.

If you have already entered a team file, you should perform the read function first to have all the players' names and numbers in memory each time you run the program. If this is the first time you are entering data for a team, you should perform the write function first. Obviously, you can have several files with information about different teams so that whenever you want to enter data for a particular team, you can easily recall from a file that team's names and numbers.

Notice in line 710 the **STR$** function, which does the opposite of the **VAL** function discussed earlier. **STR$** causes the numeric value of PN to be changed into the string value PN$. This number can then be joined together into one string with the name (PNAME$) and written ("printed") to the file as a single string or group of characters (line 810). As in the read routine

```
 780 IF LENPN>20 THEN 760
 790 IF LENPN=20 THEN 800
 800 PNAME$=PNAME$&SEG$(B$,1,20-LENPN)
 810 PRINT #1:PN$&PNAME$
 820 GOTO 690
 830 PRINT #1:"END"
 840 CLOSE #1
 850 PRINT "END OF PLAYER WRITE"
 860 GOTO 160
 870 REM ********************
 880 INPUT "PLAYER ##: ":PL
 890 IF PL=999 THEN 160
 900 IF PL>99 THEN 880
 910 INPUT "MONTH: ":MO
 920 IF MO<1 THEN 840
 930 IF MO>12 THEN 840
 940 INPUT "DAY: ":DY
 950 IF DY<1 THEN 940
 960 IF DY>31 THEN 940
 970 INPUT "YEAR: ":YR
 980 INPUT "OPPONENT: ":O$
 990 LENO=LEN(O$)
1000 IF LENO>8 THEN 980
1010 INPUT "POINTS: ":PT
1020 INPUT "FG: ":FG
1030 INPUT "FGA: ":FGA
1040 INPUT "FT: ":FT
1050 INPUT "FTA: ":FTA
1060 INPUT "REBS: ":REB
1070 INPUT "ASSISTS: ":A
1080 INPUT "STEALS: ":ST
1090 INPUT "MINUTES: ":MIN1
1100 INPUT "PF: ":PF
1110 INPUT "ACCEPT (Y/N): ":YN$
1120 IF YN$<>"Y" THEN 880
1130 PL$=STR$(PL)
1140 IF PL>9 THEN 1160
1150 PL$="0"&PL$
1160 MO$=STR$(MO)
1170 IF MO>9 THEN 1190
```

discussed above, file #1 in lines 650, 680, 810, 830, and 840 is either a cassette or a disk file. The writing is terminated when **999** is entered for the player number (line 700).

Game details for players are entered in lines 880-1640. Again, the entry is terminated by entering player number **999** in line 880. Data to be entered includes game date, opponent (limited to an eight-character abbreviation), and the player's statistics (points, field goals, field goals attempted, free throws, free throws attempted, rebounds, assists, steals, minutes played, and personal fouls).

This data is then strung together in a set sequence so that the details can be easily sorted. Each field that was keyed is changed to a string of a specified length. For example, lines 1230-1300 make the points field three digits long, even if fewer than 10 points were scored. Then all of these string values are joined together into the next available detail array element (D$). In this manner, if you have already input a file of details (see the next paragraph), each detail you key will be added to the data already read in. Line 1620 keeps track of what position within the array is available for the next detail entry.

To input a file of details previously saved to tape or diskette, lines 1660-1830 are used. Another feature is that you don't have to clear the details already in memory just because you want to read in a new details file. You can merge two files together in memory (the D$ array), then write them out to a single file. Likewise, you can read in one or more files and also add more keyed details before writing them out to a file.

```
1180 MO$="0"&MO$
1190 DY$=STR$(DY)
1200 IF DY>9 THEN 1220
1210 DY$="0"&DY$
1220 YR$=STR$(YR)
1230 REM POINTS FIELD IS 3 DIGITS
1240 PT$=STR$(PT)
1250 IF PT<10 THEN 1280
1260 IF PT<100 THEN 1300
1270 GOTO 1310
1280 PT$="00"&PT$
1290 GOTO 1310
1300 PT$="0"&PT$
1310 FG$=STR$(FG)
1320 IF FG>9 THEN 1340
1330 FG$="0"&FG$
1340 FGA$=STR$(FGA)
1350 IF FGA>9 THEN 1370
1360 FGA$="0"&FGA$
1370 FT$=STR$(FT)
1380 IF FT>9 THEN 1400
1390 FT$="0"&FT$
1400 FTA$=STR$(FTA)
1410 IF FTA>9 THEN 1430
1420 FTA$="0"&FTA$
1430 REB$=STR$(REB)
1440 IF REB>9 THEN 1460
1450 REB$="0"&REB$
1460 A$=STR$(A)
1470 IF A>9 THEN 1490
1480 A$="0"&A$
1490 ST$=STR$(ST)
1500 IF ST>9 THEN 1520
1510 ST$="0"&ST$
1520 MIN$=STR$(MIN1)
1530 IF MIN1>9 THEN 1550
1540 MIN$="0"&MIN$
1550 PF$=STR$(PF)
1560 O2$=""
1570 LENO2=8-LENO
```

Note that when the string "END" is read from the file, the input sequence is terminated (line 1780). You can, of course, clear all previous details in memory by responding **Y** in line 1660. This has the effect of resetting the counter ECTR to 0, instead of continuing from the value it had reached from a previous file read or from a keying of details.

Lines 1850-1980 save all the data details to either a diskette or a tape file. Notice that in line 1910 you have complete control over the file name you want for data files. This is true in all cases where you request to read or write a disk file. For a cassette file, you are simply asked to rewind the tape, hit play/record (or play), etc., as described in the TI manual for the operation of cassette recorders.

The sorting of player details is a "bubble sort," which goes through the array D$ and swaps the "position prior" to the "position after" if the data is greater than that of the following position. With this method the data elements "bubble" from one end of the array to the end where they belong. If any one element is detected out of order and is switched, the check through the array is done again until no swaps are made and the data is in the desired sequence. Note that only the first eight characters (which contain the player number and the date of the game) are compared. Line 2130 checks for a change having been made, and lines 2080-2100 accomplish the "swap."

A simple data listing is the result of lines 2170-2270. The data is displayed either on the screen or on both the screen and a

```
1580 O3$=O$&SEG$(O2$,1,LENO2)
1590 REC$=PL$&YR$&MO$&DY$&O3$&PT$&FG$
1600 REC2$=FGA$&FT$&FTA$&REB$&A$
1610 REC3$=ST$&MIN$&PF$
1620 ECTR=ECTR+1
1630 D$(ECTR)=REC$&REC2$&REC3$
1640 GOTO 880
1650 REM ******************
1660 INPUT "CLEAR OLD DETAILS (Y/N): ":CYN$
1670 IF CYN$<>"Y" THEN 1690
1680 ECTR=0
1690 INPUT "DISK/CASSETTE (D/C): ":DC$
1700 IF DC$="C" THEN 1730
1710 IF DC$="D" THEN 1750
1720 GOTO 1690
1730 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
1740 GOTO 1770
1750 INPUT "DSK1.FILENAME: ":FILE1$
1760 OPEN #1:FILE1$,INPUT
1770 INPUT #1:D1$
1780 IF D1$="END" THEN 1820
1790 ECTR=ECTR+1
1800 D$(ECTR)=D1$
1810 GOTO 1770
1820 CLOSE #1
1830 GOTO 160
1840 REM ******************
1850 INPUT "DISK/CASSETTE (D/C): ":DC$
1860 IF DC$="C" THEN 1890
1870 IF DC$="D" THEN 1910
1880 GOTO 1850
1890 OPEN #1:"CS1",OUTPUT,INTERNAL,
     FIXED 128
1900 GOTO 1930
1910 INPUT "DSK1.FILENAME: ":FILE1$
1920 OPEN #1:FILE1$,OUTPUT
1930 FOR I=1 TO ECTR
1940 PRINT #1:D$(I)
1950 NEXT I
```

(Continued in next column)

printer in the form in which the data is stored in the detail array. If a sort has been done, the data will be in player number/date sequence. Otherwise, the data will be in the order entered and/or read from a file or files.

Lines 2290-3030 print formatted details and totals, including averages for each player, if desired. You can make the display of data on the screen pause by answering **Y** to the input statement in line 2300. **GOSUB** 3540 in line 2340 prints a heading on the printer. **GOSUB** 3340 in line 2360 zeroes out the player totals at the beginning and after a player's last detail has been read.

LASTPL$ in line 2350 keeps track of the last player number of the previous record read from the array. When that number changes, the totals of the previous player are computed and printed. Lines 2370-2630 "unstring" the data into its elements so that totals can be added. The **SEG$** function is used to break up the large detail string array. For example, rebounds are in the 28th position of the D$ field for a length of two positions (see line 2480). A player summary is printed by the subroutine 3050-3330.

(Continued from previous column)

```
1960 PRINT #1:"END"
1970 CLOSE #1
1980 GOTO 160
1990 REM ******************
2000 REM SORT BY ##/BY DATE
```

```
2010 SORTCTR=0
2020 CHGFLAG$="N"
2030 SORTCTR=SORTCTR+1
2040 CALL CLEAR
2050 PRINT "SORT PASS # ";SORTCTR
2060 FOR I=1 TO ECTR-1
2070 IF SEG$(D$(I),1,8)<=SEG$(D$(I+1),1,8)
     THEN 2120
2080 SAV$=D$(I)
2090 D$(I)=D$(I+1)
2100 D$(I+1)=SAV$
2110 CHGFLAG$="Y"
2120 NEXT I
2130 IF CHGFLAG$="Y" THEN 2020
2140 PRINT "SORT COMPLETED"
2150 GOTO 160
2160 REM ******************
2170 INPUT "PRINTER (Y/N): ":PYN$
2180 IF PYN$<>"Y" THEN 2200
2190 OPEN #2:"RS232.BA=1200"
2200 FOR I=1 TO ECTR
2210 PRINT I;D$(I)
2220 IF PYN$<>"Y" THEN 2240
2230 PRINT #2:I;D$(I)
2240 NEXT I
2250 IF PYN$<>"Y" THEN 2270
2260 CLOSE #2
2270 GOTO 160
2280 REM ******************
2290 INPUT "PRINTER (Y/N): ":PYN$
2300 INPUT "PAUSE (Y/N): ":PAYN$
2310 INPUT "PRINT TOTALS (Y/N):":TOTYN$
2320 IF PYN$<>"Y" THEN 2350
2330 OPEN #2:"RS232.BA=1200"
2340 GOSUB 3540
2350 LASTPL$="XX"
2360 GOSUB 3340
2370 FOR I=1 TO ECTR
2380 PL$=SEG$(D$(I),1,2)
```

(Continued in next column)

(Continued from previous column)

```
2390 YR$=SEG$(D$(I),3,2)
2400 MO$=SEG$(D$(I),5,2)
2410 DY$=SEG$(D$(I),7,2)
2420 O$=SEG$(D$(I),9,8)
2430 PT$=SEG$(D$(I),17,3)
2440 FG$=SEG$(D$(I),20,2)
2450 FGA$=SEG$(D$(I),22,2)
2460 FT$=SEG$(D$(I),24,2)
2470 FTA$=SEG$(D$(I),26,2)
2480 REB$=SEG$(D$(I),28,2)
2490 A$=SEG$(D$(I),30,2)
2500 ST$=SEG$(D$(I),32,2)
2510 MIN$=SEG$(D$(I),34,2)
2520 PF$=SEG$(D$(I),36,2)
2530 PL=VAL(PL$)
2540 PT=VAL(PT$)
2550 FG=VAL(FG$)
2560 FGA=VAL(FGA$)
2570 FT=VAL(FT$)
2580 FTA=VAL(FTA$)
2590 REB=VAL(REB$)
2600 A=VAL(A$)
2610 ST=VAL(ST$)
2620 MIN1=VAL(MIN$)
2630 PF=VAL(PF$)
2640 IF PL$=LASTPL$ THEN 2700
2650 IF LASTPL$="XX" THEN 2700
2660 IF TOTYN$<>"Y" THEN 2700
2670 REM *PLAYER SUMMARY*
2680 GOSUB 3040
2690 GOSUB 3340
2700 TOTPT=TOTPT+PT
2710 TOTFG=TOTFG+FG
2720 TOTFGA=TOTFGA+FGA
2730 TOTFT=TOTFT+FT
2740 TOTFTA=TOTFTA+FTA
2750 TOTREB=TOTREB+REB
2760 TOTA=TOTA+A
2770 TOTST=TOTST+ST
```

```
2780 TOTMIN=TOTMIN+MIN1
2790 TOTG=TOTG+1
2800 LASTPL$=PL$
2810 TOTPF=TOTPF+PF
2820 IF PYN$<>"Y" THEN 2880
2830 PRINT #2:PL$;" ";P$(PL);TAB(24);MO$;"/";
     DY$;"/";YR$;
2840 PRINT #2:TAB(33);O$;TAB(42);PT$;TAB(47);
     REB$;
2850 PRINT #2:TAB(51);A$;TAB(55);ST$;TAB(59);
     MIN$;
2860 PRINT #2:TAB(63);PF$;TAB(66);FG$;"/";FGA$;
2870 PRINT #2:TAB(74);FT$;"/";FTA$
2880 PRINT "# ";PL$;" ";P$(PL)
2890 PRINT "DATE ";MO$;"/";DY$;"/";YR$
2900 PRINT "OPP: ";O$
2910 PRINT "PTS: ";PT$;" REBS: ";REB$;" A: ";A$
2920 PRINT "ST: ";ST$;" MIN: ";MIN$;" PF: ";PF$
2930 PRINT "FG/FGA: ";FG$;"/";FGA$;
     "    FT/FTA: ";FT$;"/";FTA$
2940 PRINT "**************************"
2950 IF PAYN$<>"Y" THEN 2970
2960 INPUT " ":X$
2970 NEXT I
2980 IF TOTYN$<>"Y" THEN 3000
2990 GOSUB 3040
3000 INPUT "HIT ENTER TO CONTINUE ":X$
3010 IF PYN$<>"Y" THEN 3030
3020 CLOSE #2
3030 GOTO 160
3040 REM **************************
3050 FGP=INT(TOTFG*100000/TOTFGA)
3060 PPG=INT(TOTPT/TOTG*100)/100
3070 RPG=INT(TOTREB/TOTG*100)/100
3080 MPG=INT(TOTMIN/TOTG*100)/100
3090 FTP=INT(TOTFT*100000/TOTFTA)
3100 FTP=FTP/1000
3110 FGP=FGP/1000
3120 PRINT "*PLAYER TOTALS* #GAMES:";TOTG
```

(Continued in next column)

(Continued from previous column)

```
3130 PRINT "PTS:";TOTPT;"REBS:";TOTREB;"A:";
     TOTA
3140 PRINT "ST:";TOTST;"MIN:";TOTMIN;"PF:";
     TOTPF
3150 PRINT "FG/FGA:";TOTFG;"/";TOTFGA;
     TAB(20);FGP;"%"
3160 PRINT "FT/FTA:";TOTFT;"/";TOTFTA;
     TAB(20);FTP;"%"
3170 PRINT "PTS/GM:";PPG
3180 PRINT "MIN/GM:";MPG
3190 PRINT "REB/GM:";RPG
3200 PRINT "**************************"
3205 IF PYN$<>"Y" THEN 3330
3210 PRINT #2
3220 PRINT #2:"*PLAYER TOTALS* #GAMES:";
     TOTG
3230 PRINT #2:"PTS:";TOTPT;" REB:";TOTREB;
     "  A:";TOTA;
3240 PRINT #2:" ST:";TOTST;" MIN:";TOTMIN;
     "  PF:";TOTPF
3250 PRINT #2:"FG/FGA:";TOTFG;"/";TOTFGA;
     TAB(20);FGP;"%";TAB(40);
3260 PRINT #2:"FT/FTA:";TOTFT;"/";TOTFTA;
     TAB(60);FTP;"%"
3270 PRINT #2:"PTS/GM:";PPG;TAB(21);
3280 PRINT #2:"MIN/GM:";MPG;TAB(40);
3290 PRINT #2:"REB/GM:";RPG
3300 PRINT #2:"**************************";
3310 PRINT #2:"**************************";
3320 PRINT #2:"**************************"
3330 RETURN
3340 TOTFGA=0
3350 TOTFT=0
3360 TOTFTA=0
3370 TOTREB=0
3380 TOTA=0
3390 TOTST=0
3400 TOTMIN=0
3410 TOTPF=0
```

```
3420 TOTG=0
3430 TOTPT=0
3440 TOTFG=0
3450 RETURN
3460 REM ******************
3470 ECTR=0
3480 PRINT "DETAILS CLEARED"
3490 GOTO 160
3500 FOR I=0 TO 99
3510 P$(I)=" "
3520 NEXT I
3530 RETURN
3540 PRINT #2:"## NAME                ";
3545 PRINT #2:"  DATE      "
3550 PRINT #2:"       PTS  REB A   ST  MIN ";
3560 PRINT #2:"PF FG/FGA  FT/FTA"
3570 PRINT #2
3580 RETURN
3590 END
```

```
## NAME                  DATE      OPP       PTS  REB A   ST  MIN PF FG/FGA   FT/FTA

08 JAY K.                03/12/83 RED       014  08  02  01  12  3  06/09    02/03
08 JAY K.                03/20/83 BLUE      020  04  02  00  13  2  08/13    04/06

*PLAYER TOTALS* #GAMES: 2
PTS: 34   REB: 12   A: 4  ST: 1   MIN: 25   PF: 5
FG/FGA: 14 / 22     63.636 %          FT/FTA: 6 / 9          66.666 %
PTS/GM: 17          MIN/GM: 12.5      REB/GM: 6
*******************************************************************************
12 DEREK                 03/12/83 RED       015  07  02  02  13  4  04/05    07/10
12 DEREK                 03/20/83 BLUE      018  06  01  01  10  1  09/10    00/01

*PLAYER TOTALS* #GAMES: 2
PTS: 33   REB: 13   A: 3  ST: 3   MIN: 23   PF: 5
FG/FGA: 13 / 15     86.666 %          FT/FTA: 7 / 11         63.636 %
PTS/GM: 16.5        MIN/GM: 11.5      REB/GM: 6.5
*******************************************************************************
```

```
15 JAY H.              03/12/83 RED      020  06  04  03  14  5  10/15   00/02
15 JAY H.              03/20/83 BLUE     010  08  01  00  10  2  03/06   04/07

*PLAYER TOTALS* #GAMES: 2
PTS: 30   REB: 14   A: 5  ST: 3   MIN: 24   PF: 7
FG/FGA: 13 / 21     61.904 %          FT/FTA: 4 / 9         44.444 %
PTS/GM: 15          MIN/GM: 12        REB/GM: 7
*******************************************************************************
19 VAUGHN              03/12/83 RED      012  06  01  01  12  2  05/09   02/02

*PLAYER TOTALS* #GAMES: 1
PTS: 12   REB: 6    A: 1  ST: 1   MIN: 12   PF: 2
FG/FGA: 5 / 9       55.555 %          FT/FTA: 2 / 2         100 %
PTS/GM: 12          MIN/GM: 12        REB/GM: 6
*******************************************************************************
24 RICK                03/12/83 RED      016  09  03  02  13  2  06/12   04/07
24 RICK                03/20/83 BLUE     006  06  00  01  10  5  03/03   00/01

*PLAYER TOTALS* #GAMES: 2
PTS: 22   REB: 15   A: 3  ST: 3   MIN: 23   PF: 7
FG/FGA:  9 / 15     60 %              FT/FTA: 4 / 8         50 %
PTS/GM: 11          MIN/GM: 11.5      REB/GM: 7.5
*******************************************************************************
29 BOBBY               03/12/83 RED      008  10  04  03  08  3  04/07   00/00
29 BOBBY               03/20/83 BLUE     021  12  01  01  16  2  09/10   03/03

*PLAYER TOTALS* #GAMES: 2
PTS: 29   REB: 22   A: 5  ST: 4   MIN: 24   PF: 5
FG/FGA: 13 / 17     76.47 %           FT/FTA: 3 / 3         100 %
PTS/GM: 14.5        MIN/GM: 12        REB/GM: 11
*******************************************************************************
33 TONY                03/12/83 RED      010  04  01  00  09  3  04/05   02/02
33 TONY                03/20/83 BLUE     009  01  03  04  12  2  04/05   01/01

*PLAYER TOTALS* #GAMES: 2
PTS: 19   REB: 5    A: 4  ST: 4   MIN: 21   PF: 5
FG/FGA: 8 / 10      80 %              FT/FTA: 3 / 3         100 %
PTS/GM: 9.5         MIN/GM: 10.5      REB/GM: 2.5
*******************************************************************************
```

CALC1

```
100 REM * ************************
110 REM *          CALC1          *
120 REM *                         *
130 REM *     COPYRIGHT 1983      *
140 REM *   DONALD C. KREUTNER    *
150 REM * ************************
160 REM
170 REM
180 TOT=0
190 CALL CLEAR
200 INPUT A$
210 LENA=LEN(A$)
220 IF LENA=0 THEN 200
230 IF A$="END" THEN 1250
240 IF A$<>"C" THEN 280
250 TOT=0
260 CALL CLEAR
270 GOTO 200
280 NUM1$=" "
290 NUM2$=" "
300 OPER1$=" "
310 OPER2$=" "
320 NUM1CTR=0
330 NUM2CTR=0
340 IF SEG$(A$,1,1)>="0" THEN 360
350 GOTO 370
360 IF SEG$(A$,1,1)<="9" THEN 390
370 FLAGOPER1$="Y"
380 GOTO 400
390 FLAGOPER1$="N"
400 FOR I=1 TO LENA
410 A2$=SEG$(A$,I,1)
420 IF A2$="X" THEN 700
430 IF A2$="/" THEN 700
440 IF A2$="^" THEN 700
450 IF A2$="+" THEN 700
460 IF A2$="-" THEN 700
```

CALC1 performs like a calculator—adding, subtracting, multiplying, dividing, and taking the powers and roots of various numbers. The total accumulator (TOT) is cleared to zero by line 180 at the program startup. Every calculation to be performed is entered in line 200. Whatever you key in is then evaluated by lines 220 to the program end. After every evaluation, control is returned to line 200 for the next operation.

Two numbers can be entered, which are separated by an operator and terminated by an equal sign, as in

243.5/12.2=

When a number starts the keyed expression, the previously accumulated total (TOT) is automatically cleared. This is determined in lines 340 to 390, where the first character entered is inspected to see whether it is numeric or an operator.

Lines 420-550 ensure that the operation is valid (either **X**, **/**, +, **-**, ^, **A**, **S**, a period, or a number).

Then, depending on whether the number being read one character at a time is the first or the second number, lines 560-690 move the numbers and/or decimal points into NUM1$ or NUM2$. The string entered is examined for the length of the string (LENA in line 400). When that has been completed, NUM1 and NUM2 are given numeric values from the strings NUM1$ and NUM2$ (lines 770 and 800). Lines 810 and 820 clear the total if the first character keyed is numeric. Lines 830-1020 then

```
470 IF A2$="A" THEN 700
480 IF A2$="S" THEN 700
490 IF A2$="=" THEN 700
500 IF A2$="." THEN 560
510 IF A2$>"9" THEN 540
520 IF A2$<"0" THEN 540
530 GOTO 560
540 PRINT "MUST BE # OR X,/,^,+,-,A,S,="
550 GOTO 200
560 IF OPER1$=" " THEN 640
570 IF FLAGOPER1$="Y" THEN 640
580 NUM2CTR=NUM2CTR+1
590 IF NUM2CTR>1 THEN 620
600 NUM2$=SEG$(A2$,1,1)
610 GOTO 630
620 NUM2$=NUM2$&SEG$(A2$,1,1)
630 GOTO 740
640 NUM1CTR=NUM1CTR+1
650 IF NUM1CTR>1 THEN 680
660 NUM1$=SEG$(A2$,1,1)
670 GOTO 690
680 NUM1$=NUM1$&SEG$(A2$,1,1)
690 GOTO 740
700 IF OPER1$=" " THEN 730
710 OPER2$=A2$
720 GOTO 740
730 OPER1$=A2$
740 NEXT I
750 IF LEN(NUM1$)<1 THEN 780
760 IF NUM1$=" " THEN 780
770 NUM1=VAL(NUM1$)
780 IF LEN(NUM2$)<1 THEN 810
790 IF NUM2$=" " THEN 810
800 NUM2=VAL(NUM2$)
810 IF FLAGOPER1$="Y" THEN 1030
820 TOT=0
830 IF OPER1$="A" THEN 910
840 IF OPER1$="+" THEN 910
850 IF OPER1$="S" THEN 930
860 IF OPER1$="-" THEN 930
```

compute the total to be the result of NUM1 and NUM2, acted on by the first operator. Then, if the second operator is an equal sign, the result is printed (lines 1000 and 1010). However, if the first character is an operator, the previous total is added to, multiplied, etc., to obtain a new total. Once again, whenever an equal sign is encountered, the total is displayed on the screen.

Following are statements of particular interest.

Line 410 evaluates the characters entered one character at a time. **SEG$**(A$,I,1) represents the segment of A$, starting at position number I for a length of one character. Note that I increases by one as the **FOR-NEXT** loop of lines 400-740 is executed.

Line 210 determines the length of A$ so that it can be evaluated character by character. The function **LEN**(A$) determines the length of A$. Finally, remember that **A** and **S** can be used more efficiently than the operators + and -, since these require the holding down of the shift key.

```
89.5X16 =
1432
/ 2 =
716
14.52-6.2
-5
=
3.32
C
X10 =
0
```

```
870 IF OPER1$="X" THEN 950
880 IF OPER1$="/" THEN 970
890 IF OPER1$="^" THEN 990
900 GOTO 200
910 TOT=NUM1+NUM2
920 GOTO 1000
930 TOT=NUM1-NUM2
940 GOTO 1000
950 TOT=NUM1*NUM2
960 GOTO 1000
970 TOT=NUM1/NUM2
980 GOTO 1000
990 TOT=NUM1^NUM2
1000 IF OPER2$<>"=" THEN 200
1010 PRINT "*TOTAL*";TOT
1020 GOTO 200
1030 IF OPER1$="=" THEN 1230
1040 IF OPER1$="+" THEN 1120
1050 IF OPER1$="A" THEN 1120
1060 IF OPER1$="-" THEN 1140
1070 IF OPER1$="S" THEN 1140
1080 IF OPER1$="X" THEN 1160
1090 IF OPER1$="/" THEN 1180
1100 IF OPER1$="^" THEN 1200
1110 GOTO 200
1120 TOT=TOT+NUM1
1130 GOTO 1210
1140 TOT=TOT-NUM1
1150 GOTO 1210
1160 TOT=TOT*NUM1
1170 GOTO 1210
1180 TOT=TOT/NUM1
1190 GOTO 1210
1200 TOT=TOT^NUM1
1210 IF OPER2$="=" THEN 1230
1220 GOTO 200
1230 PRINT "*TOTAL*";TOT
1240 GOTO 200
1250 END
```

CALENDAR

```
100 REM * ************************ *
110 REM *                          *
120 REM *          CALENDAR        *
130 REM *                          *
140 REM *       COPYRIGHT 1983     *
150 REM *     DONALD C. KREUTNER   *
160 REM * ************************ *
170 REM CALENDAR
180 REM 1980 IS BASE FOR LY/3 (TUE)
190 REM 1981 IS BASE FOR LY+1/5 (THU)
200 REM 1982 IS BASE FOR LY+2/6 (FRI)
210 REM 1983 IS BASE FOR LY+3/7 (SAT)
220 REM 01/01/80 FELL ON TUE, 3RD DAY OF
    WEEK
230 INPUT "DESIRED 4 DIGIT YR FOR
    CALENDAR: ":Y1
235 IF Y1=0 THEN 2110
240 IF Y1<1800 THEN 230
250 IF Y1>2099 THEN 230
270 INTYRDIV4=INT(Y1/4)
280 LY1=INTYRDIV4*4
290 BASE1=Y1-LY1
300 REM BASE1=0 (LEAP YEAR)
310 REM BASE1=1 (LY + 1)
320 REM BASE1=2 (LY + 2)
330 REM BASE1=3 (LY + 3)
340 REM 1800 & 1900 ARE NOT LY'S
350 IF Y1<1900 THEN 390
360 IF Y1<2000 THEN 490
370 IF Y1<2100 THEN 580
380 REM ******************
390 BEGY=1801
400 ENDY=1899
410 IF Y1<>1800 THEN 450
420 REM1=4
430 BASE1=1
440 GOTO 900
```

CALENDAR produces a calendar for any year from 1800 to 2099. With this program you can find anyone's birth date to the exact day of the week or print out any number of calendars for any year.

The logic behind the calculation is briefly explained in the remarks of lines 180-220. Each day of the week is given a number (Sundays are 1, Mondays are 2, and so on, through Saturdays, which are 7.) Each succeeding year has January 1st falling one day number greater than the day number of January 1st for the previous year. The only exceptions to this rule are the following:

1. Two days are added on the year following a leap year.
2. Not all century years are leap years even though they are divisible by four.

These "centesimal" years, which end with two zeros, are only leap years if they are evenly divisible by 400. Hence, 1800 and 1900 were not leap years, whereas the year 2000 will be.

As noted, this program computes calendars for only the years from 1800 to 2099. If you want to extend the upper and lower ranges beyond these limits, lines 390-480 set up the parameters for the 1800s, lines 490-570 for the 1900s, and lines 580-660 for the 21st century. To add other centuries, you need only add similar modules for the beginning and ending years. Note that REM1 being 4 in line 420 indicates that

```
450 DAY1=4
460 LYF=99
470 REM **LYF=99 MEANS COMMON CENTURY
    YR**
480 GOTO 670
490 BEGY=1901
500 ENDY=1999
510 IF Y1<>1900 THEN 550
520 REM1=2
530 BASE1=1
540 GOTO 900
550 DAY1=2
560 LYF=99
570 GOTO 670
580 BEGY=2001
590 ENDY=2099
600 IF Y1<>2000 THEN 640
610 REM1=7
620 BASE1=0
630 GOTO 900
640 DAY1=7
650 LYF=0
660 GOTO 670
670 FOR I=BEGY TO ENDY
680 LYF=LYF+1
690 IF LYF>90 THEN 810
700 IF LYF<4 THEN 770
710 LYF=0
720 REM LYF=0 MEANS LY
730 REM LYF=1 MEANS LY + 1
740 REM LYF=2 MEANS LY + 2
750 REM LYF=3 MEANS LY + 3
760 REM LYF=99 MEANS COMMON CENTURY
    YR
770 IF LYF<>1 THEN 810
780 DAY1=DAY1+2
790 GOTO 840
800 DAY1=5
810 DAY1=DAY1+1
820 IF LYF<90 THEN 840
```

January 1st in 1800 fell on Wednesday. BASE1 in line 430 is 1, meaning that 1800 was not a leap year. And LYF (leap year flag) is 99, which means that 1800 was a nonleap (or common) century year.

From this basic information, by counting from 1800, 1900, or 2000 to the desired calendar year, you can easily calculate exactly what day January 1st was in any year and whether or not that year was a leap year. The generation of a calendar for the year is then a simple matter.

Now let's discuss some specific program statements. As mentioned earlier, lines 230-660 set up the desired parameters for the first year of the century for the year a calendar was requested. You will note that you can end the program by keying **0** for the desired year in line 230. Obviously, this program, as any other, can also be terminated by holding down the **FCTN** key and depressing **CLEAR** (**4**) at the same time.

Lines 670-890 contain a **FOR-NEXT** loop that computes the day of the week for January 1st of the year asked for. The loop starts (line 670) with BEGY—defined earlier as 1800, 1900, or 2000—and ends when the year being computed equals the requested year (line 880 when the counter I = Y1, the keyed year in line 230). DAY1 (the day of the week of the first day of the year) is added to, one by one, as each year is counted off, except when the previous year was a leap year. In that event, two is added to DAY 1 (lines 770-780). Whenever DAY1 exceeds 7, the value is changed to DAY1 minus 7 (lines 840-850).

```
830 LYF=1
840 IF DAY1<8 THEN 860
850 DAY1=DAY1-7
860 REM1=DAY1
870 REM PRINT #1:I;"LYF=";LYF;"DAY1=";DAY1
880 IF I=Y1 THEN 900
890 NEXT I
900 REM REM1 IS REMAINDER (DAY FOR
    JAN 01)
910 REM CLOSE #1
920 REM1=REM1-1
930 IF REM1>0 THEN 950
940 REM1=7
950 REM PRINT "REM1=";REM1
960 INPUT "PRINTER (Y/N): ":X$
970 IF X$<>"Y" THEN 990
980 OPEN #1:"RS232.BA=1200"
990 CALL CLEAR
1000 FOR I=1 TO 12
1010 PRINT
1020 MON$="    "
1030 IF I=1 THEN 1150
1040 IF I=2 THEN 1170
1050 IF I=3 THEN 1190
1060 IF I=4 THEN 1210
1070 IF I=5 THEN 1230
1080 IF I=6 THEN 1250
1090 IF I=7 THEN 1270
1100 IF I=8 THEN 1290
1110 IF I=9 THEN 1310
1120 IF I=10 THEN 1330
1130 IF I=11 THEN 1350
1140 IF I=12 THEN 1370
1150 MON$="JAN"
1160 GOTO 1380
1170 MON$="FEB"
1180 GOTO 1380
1190 MON$="MAR"
1200 GOTO 1380
1210 MON$="APR"
```

Note that in lines 870 and 910 the residue of some "debugging" statements still exists within the program. In this case the values of LYF and DAY1 were printed to the printer to determine what was going on within the program logic. Another sometimes useful method is to say **TRACE** to the > prompt of BASIC prior to running your program. This, coupled with **PRINT** statements (to the screen), will help show you the order of program statements being executed and the values of some key variables you may be concerned about (using the **PRINT** statements).

Once the day of the week of the first day of the year has been determined and whether or not the selected year is a leap year, the program is ready to generate a calendar. Lines 1000-2080 contain two loops, an I loop counting from 1 to 12 for the months, and a J loop counting from 1 to 31 for the days of each month. Lines 1030-1370 give the month abbreviations for the months being displayed. Lines 960-980 cause the calendar to be printed on a printer as well as be displayed on the screen. CALENDAR, like all other programs, can be used with or without a printer. Obviously, the program is more useful if you own a printer.

Lines 1540-1720 determine how many days each month has (from the famous "30 days hath September . . . ." rhyme). Likewise, the day of the week is increased by 1 (from 1 to 7), and the number of the day of the month is printed in the proper day of the week column by tabbing (lines 1760-2010). Line 1760 uses the **ON-GOTO** statement, which causes a branch to the desired line when REM1 = 1, 2, 3, 4, 5, 6, or 7. When the

```
1220 GOTO 1380
1230 MON$="MAY"
1240 GOTO 1380
1250 MON$="JUN"
1260 GOTO 1380
1270 MON$="JUL"
1280 GOTO 1380
1290 MON$="AUG"
1300 GOTO 1380
1310 MON$="SEP"
1320 GOTO 1380
1330 MON$="OCT"
1340 GOTO 1380
1350 MON$="NOV"
1360 GOTO 1380
1370 MON$="DEC"
1380 PRINT " ";MON$;" /";Y1
1390 PRINT " -------------------------"
1400 PRINT " SU  MO  TU  WE  ";
1410 PRINT " TH  FR  SA  "
1420 IF X$<>"Y" THEN 1520
1430 IF I<>7 THEN 1470
1440 FOR K=1 TO 10
1450 PRINT #1
1460 NEXT K
1470 PRINT #1
1480 PRINT #1:" ";MON$;" /";Y1
1490 PRINT #1:" -------------------------"
1500 PRINT #1:" SU  MO  TU  WE  ";
1510 PRINT #1:" TH  FR  SA  "
1520 FOR J=1 TO 31
1530 IF J<28 THEN 1730
1540 IF I=1 THEN 1700
1550 IF I=3 THEN 1700
1560 IF I=5 THEN 1700
1570 IF I=7 THEN 1700
1580 IF I=8 THEN 1700
1590 IF I=10 THEN 1700
1600 IF I=12 THEN 1700
1610 IF I=9 THEN 1720
```

calendar has been completed, the printer file is closed (if it was used), and control returns to line 230 from line 2100.

```
JAN / 1984
SU  MO  TU  WE  TH  FR  SA
1   2   3   4   5   6   7
8   9   10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30  31

FEB / 1984
SU  MO  TU  WE  TH  FR  SA
                1   2   3   4
5   6   7   8   9   10  11
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29

MAR / 1984
SU  MO  TU  WE  TH  FR  SA
                    1   2   3
4   5   6   7   8   9   10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27  28  29  30  31

APR / 1984
SU  MO  TU  WE  TH  FR  SA
1   2   3   4   5   6   7
8   9   10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30

MAY / 1984
SU  MO  TU  WE  TH  FR  SA
        1   2   3   4   5
6   7   8   9   10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28  29  30  31
```

```
1620 IF I=4 THEN 1720
1630 IF I=6 THEN 1720
1640 IF I=11 THEN 1720
1650 IF BASE1=0 THEN 1680
1660 IF J>28 THEN 2040
1670 GOTO 1730
1680 IF J>29 THEN 2040
1690 GOTO 1730
1700 IF J>31 THEN 2040
1710 GOTO 1730
1720 IF J>30 THEN 2040
1730 REM1=REM1+1
1740 IF REM1<8 THEN 1760
1750 REM1=1
1760 ON REM1 GOTO 1770,1810,1850,1890,1930,
     1970,2010
1770 PRINT J;
1780 IF X$<>"Y" THEN 1800
1790 PRINT #1:J;
1800 GOTO 2040
1810 PRINT TAB(5);J;
1820 IF X$<>"Y" THEN 1840
1830 PRINT #1:TAB(5);J;
1840 GOTO 2040
1850 PRINT TAB(9);J;
1860 IF X$<>"Y" THEN 1880
1870 PRINT #1:TAB(9);J;
1880 GOTO 2040
1890 PRINT TAB(13);J;
1900 IF X$<>"Y" THEN 1920
1910 PRINT #1:TAB(13);J;
1920 GOTO 2040
1930 PRINT TAB(17);J;
1940 IF X$<>"Y" THEN 1960
1950 PRINT #1:TAB(17);J;
1960 GOTO 2040
1970 PRINT TAB(21);J;
1980 IF X$<>"Y" THEN 2000
1990 PRINT #1:TAB(21);J;
2000 GOTO 2040
```

JUN / 1984

SU	MO	TU	WE	TH	FR	SA
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JUL / 1984

SU	MO	TU	WE	TH	FR	SA
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

AUG / 1984

SU	MO	TU	WE	TH	FR	SA
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEP / 1984

SU	MO	TU	WE	TH	FR	SA
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

OCT / 1984

SU	MO	TU	WE	TH	FR	SA
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

NOV / 1984

SU	MO	TU	WE	TH	FR	SA
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

```
2010 PRINT TAB(25);J
2020 IF X$<>"Y" THEN 2040
2030 PRINT #1:TAB(25);J
2040 NEXT J
2050 PRINT
2060 IF X$<>"Y" THEN 2080
2070 PRINT #1
2080 NEXT I
2085 IF X$<>"Y" THEN 2100
2090 CLOSE #1
2100 GOTO 230
2110 END
```

```
DEC / 1984
SU  MO  TU  WE  TH  FR  SA
                        1
2   3   4   5   6   7   8
9   10  11  12  13  14  15
16  17  18  19  20  21  22
23  24  25  26  27  28  29
30  31

JAN / 1920
SU  MO  TU  WE  TH  FR  SA
                1   2   3
4   5   6   7   8   9   10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27  28  29  30  31
```

DIVMULT1

```
100 REM ****************************
110 REM *                          *
120 REM *          DIVMULT1        *
130 REM *       COPYRIGHT 1983     *
140 REM *                          *
150 REM *     DONALD C. KREUTNER   *
160 REM ****************************
170 REM MULTDIV1
180 RANDOMIZE
190 CALL CLEAR
200 INPUT "WITH BORDER (Y/N): ":YN$
210 INPUT "TOP # FROM 1 TO (2/999): ":T1
220 IF T1<2 THEN 210
230 IF T1>999 THEN 210
240 INPUT "BOTTOM # 1 TO (2/999): ":B1
250 IF B1<2 THEN 240
260 IF B1>999 THEN 240
270 ERRS=0
280 CORR=0
290 INPUT "DIV, MULT, OR BOTH (D/M/B): ":
    DMB$
300 IF DMB$="D" THEN 340
310 IF DMB$="M" THEN 340
320 IF DMB$="B" THEN 340
330 GOTO 290
340 B$="FFFFFFFFFFFFFFFF"
350 U$="000000FFFF000000"
360 DIV$="181800FFFF001818"
370 CALL CHAR(96,B$)
380 CALL CHAR(97,U$)
390 CALL CHAR(98,DIV$)
400 CALL CLEAR
410 IF YN$="N" THEN 460
420 CALL HCHAR(8,10,96,12)
430 CALL HCHAR(18,10,96,12)
440 CALL VCHAR(9,10,96,9)
450 CALL VCHAR(9,21,96,9)
460 NUM1=INT(T1*RND)+1
```

DIVMULT1 is very similar to ADDSUB1, but the skills DIVMULT1 improves are division and multiplication. The screen layout and program logic are much the same as those described in ADDSUB1.

Again, you can have a border around the problem, if desired. You can have only multiplication or division, or both (line 290). In lines 360 and 390, one additional character, the division sign (a hyphen with a dot above and a dot below), is defined.

The top and bottom numbers are selected randomly in lines 460 and 470, as in the addition/subtraction program. In lines 480-540 the operation (division or multiplication) is determined.

The method of displaying the two numbers is the same as in ADDSUB1, moving one character at a time to the two rows in the center of the screen and asking for the answer at the bottom of the display.

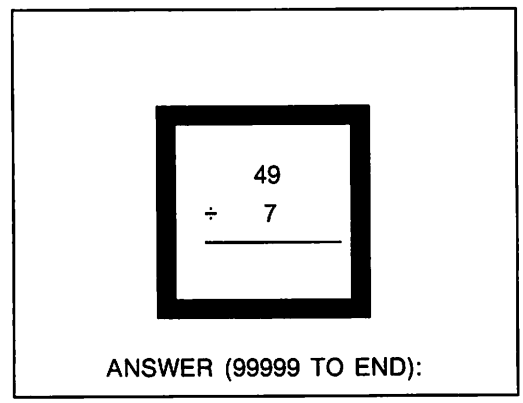
This program is particularly good for drilling a young child on the multiplication tables, since you can select the ending ranges for both the top and bottom numbers. For instance, you can start with a top value of 1 to 9 and a bottom value of 1 to 2, and then gradually work up to 1 to 12 on top and bottom, constantly reviewing already learned material every day.

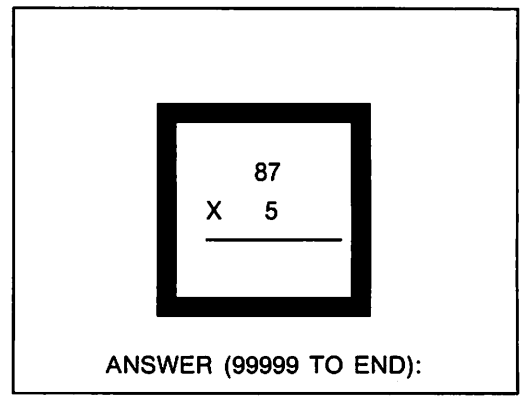
DIVMULT1 can also sharpen the skills of an adult who tries to respond as quickly and accurately as possible. And remember that

```
470 NUM2=INT(B1*RND)+1
480 IF DMB$="D" THEN 540
490 IF DMB$="M" THEN 520
500 OP1=INT(2*RND)+1
510 GOTO 550
520 OP1=1
530 GOTO 550
540 OP1=2
550 IF OP1=1 THEN 570
560 NUM1=NUM1*NUM2
570 NUM1$=STR$(NUM1)
580 NUM2$=STR$(NUM2)
590 IF OP1=1 THEN 620
600 OP$="/"
610 GOTO 630
620 OP$="X"
630 LEN1=LEN(NUM1$)
640 LEN2=LEN(NUM2$)
650 J=18
660 FOR I=LEN1 TO 1 STEP -1
670 J=J-1
680 V=ASC(SEG$(NUM1$,I,1))
690 CALL VCHAR(12,J,V,1)
700 NEXT I
710 J=18
720 FOR I=LEN2 TO 1 STEP -1
730 J=J-1
740 V=ASC(SEG$(NUM2$,I,1))
750 CALL VCHAR(13,J,V,1)
760 NEXT I
770 OP2=ASC(OP$)
780 IF OP$<>"/" THEN 800
790 OP2=98
800 CALL VCHAR(13,14,OP2,1)
810 CALL HCHAR(14,14,97,4)
820 INPUT "ANSWER (OR 0 TO END): ":X
830 IF X<=0 THEN 990
840 IF OP1=2 THEN 870
850 IF X=NUM1*NUM2 THEN 950
860 GOTO 880
```

both top and bottom numbers can range up to 999.

See the sample screens below.





```
870 IF X=NUM1/NUM2 THEN 950
880 CALL CLEAR
890 IF OP1=2 THEN 920
900 PRINT "INCORRECT";NUM1;"X";NUM2;"=";
    NUM1*NUM2
910 GOTO 930
920 PRINT "INCORRECT";NUM1;"/";NUM2;"=";
    NUM1/NUM2
930 ERRS=ERRS+1
940 GOTO 970
950 CORR=CORR+1
960 GOTO 400
970 INPUT " ":X$
980 GOTO 400
990 CALL CLEAR
1000 PRINT "************************"
1010 PRINT "*";TAB(23);"*"
1020 PRINT "*CORRECT =";CORR;TAB(23);"*"
1030 PRINT "*";TAB(23);"*"
1040 PRINT "*ERRORS=";ERRS;TAB(23);"*"
1050 PRINT "*";TAB(23);"*"
1060 IF CORR>0 THEN 1090
1070 IF ERRS>0 THEN 1090
1080 ERRS=1
1090 PRINT "*SCORE =";INT(CORR/(CORR+ERRS)
    *100);"%";TAB(23);"*"
1100 PRINT "*";TAB(23);"*"
1110 PRINT "************************"
1120 PRINT
1130 PRINT
1140 END
```

GRAPH1

```
100 REM * ************************
110 REM *                        *
120 REM *          GRAPH1        *
130 REM *       COPYRIGHT 1983   *
140 REM *                        *
150 REM *     DONALD C. KREUTNER *
160 REM * ************************
170 REM GRAPH1
180 OPTION BASE 1
190 PYN$="N"
200 CALL CHAR(96,"FF818181818181FF")
210 DIM V(10)
220 DIM N$(10)
230 CALL CLEAR
240 CTR=0
250 INPUT "MAX VALUE: ":MAX
260 IF MAX=0 THEN 830
270 INPUT "GRAPH NAME(18 CHARS): ":GN$
280 IF LEN(GN$)>18 THEN 270
290 IF PYN$="N" THEN 310
300 CLOSE #1
310 INPUT "PRINTER (Y/N): ":PYN$
320 IF PYN$="Y" THEN 350
330 IF PYN$<>"N" THEN 310
340 GOTO 360
350 OPEN #1:"RS232.BA=1200"
360 CALL CLEAR
370 FOR I=1 TO 10
380 PRINT "ELEMENT #: ";I
390 INPUT "ELEMENT TITLE (9 CHAR): ":N$(I)
400 IF LEN(N$(I))>9 THEN 390
410 IF N$(I)="END" THEN 470
420 CTR=CTR+1
430 INPUT "VALUE OF ELEMENT: ":V(I)
440 IF V(I)>MAX THEN 430
450 IF V(I)<0 THEN 430
460 NEXT I
```

GRAPH1 will draw a horizontal bar graph for any set of figures. Each bar to be drawn is given a name, which is printed to the left of the bar. The name can contain up to 9 characters (the "element title" in line 390).

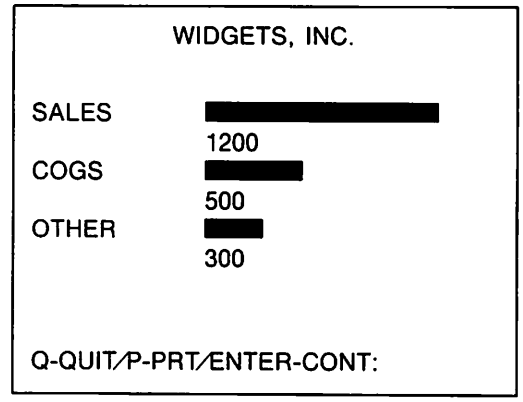
At the beginning of the program, you are asked for a "graph name," which will be the title for the graph. This title can have up to 18 characters (lines 270-280). You also need to identify the maximum value that you want the graph to be able to chart (line 250). Notice that by entering a maximum value of zero, you will end the program (line 260). The **OPTION BASE** statement in line 180 sets up the arrays V and N$ to have the lowest subscript value of 1 rather than the default of zero for TI BASIC. You can then have up to 10 bars displayed on any chart.

Once you have set up the maximum value, you begin entering values for elements 1 to the number of elements desired. You keep on entering values until you enter **END** for the element name, at which time the entry portion (lines 370-460) terminates. The maximum value previously keyed is then divided by 12 to get the value of one "square" or block in the graph (see line 470). In other words, if any element has a maximum value (they all can have lesser values or the same, but not greater), the length of that element's bar would be 12 blocks.

Lines 490-670 then display the graph on the screen and/or on the printer (if you answered **Y** in line 310). The title is printed at the top, and the name of each graph

```
470 BLOCK=INT(MAX/12)
480 CALL CLEAR
490 FOR I=1 TO CTR
500 PRINT N$(I);TAB(10);
510 IF PYN$<>"Y" THEN 530
520 PRINT #1:N$(I);TAB(10);
530 V2=INT(V(I)/BLOCK)
540 IF V2=0 THEN 600
550 FOR J=1 TO V2
560 PRINT CHR$(96);
570 IF PYN$<>"Y" THEN 590
580 PRINT #1:"O";
590 NEXT J
600 PRINT
610 PRINT TAB(10);V(I)
620 PRINT
630 IF PYN$<>"Y" THEN 670
640 PRINT #1
650 PRINT #1:TAB(10);V(I)
660 PRINT #1
670 NEXT I
680 PRINT
690 PRINT CHR$(96);" =";BLOCK;TAB(10);GN$
700 IF PYN$<>"Y" THEN 720
710 PRINT #1:"O = ";BLOCK;TAB(10);GN$
720 PRINT
730 IF PYN$<>"Y" THEN 750
740 PRINT #1
750 INPUT "Q=QUIT/P=PRT/ENTER=CONT: ":E$
760 IF E$="Q" THEN 830
770 IF E$="P" THEN 790
780 GOTO 230
790 IF PYN$="Y" THEN 480
800 OPEN #1:"RS232.BA=1200"
810 PYN$="Y"
820 GOTO 480
830 CALL CLEAR
840 IF PYN$<>"Y" THEN 860
850 CLOSE #1
860 END
```

element precedes the bar for its value. If "drawn" also on the printer, O's are used instead of the shaded blocks on the screen. When the graph is completed, the options are to quit (**Q**), print the same graph (**P**), or continue with a new graph (**ENTER**).



MORTGAGE

```
100 REM * * * * * * * * * * * * * * * * * * * * * * * *
110 REM *                MORTGAGE                *
120 REM *                                        *
130 REM *             COPYRIGHT 1983             *
140 REM *          DONALD C. KREUTNER            *
150 REM * * * * * * * * * * * * * * * * * * * * * * * *
160 CALL CLEAR
170 INPUT "PRINCIPLE:                    ":PRIN
180 IF PRIN<=0 THEN 880
190 INPUT "YEARLY INTEREST% :            ":IY
200 IY=IY/100
210 INPUT "MONTHS OF LOAN:               ":N
220 IM=IY/12
230 PAY=PRIN*IM*(1+IM)^N/(((1+IM)^N)-1)
240 PAY2=PAY*100
250 PAY3=INT(PAY2)+.5
260 IF PAY2>=PAY3 THEN 290
270 PAY2=INT(PAY2)
280 GOTO 300
290 PAY2=INT(PAY2)+1
300 PAYAMT=PAY2/100
310 PRINT "MONTHLY PAYMENT IS: ";PAYAMT
320 INPUT "DETAILED SCHEDULE (Y/N): ":YN$
330 IF YN$="N" THEN 160
340 IF YN$="Y" THEN 360
350 GOTO 320
360 INPUT "PRINTER (Y/N): ":PYN$
370 IF PYN$<>"Y" THEN 390
380 OPEN #1:"RS232.BA=1200"
390 INPUT "CALCULATED AMOUNT (Y/N): ":YN$
400 IF YN$="Y" THEN 440
410 IF YN$="N" THEN 430
420 GOTO 390
430 INPUT "DESIRED MONTHLY AMOUNT: ":
    PAYAMT
440 IF PYN$<>"Y" THEN 520
450 PRINT #1:"PRINCIPLE: ";PRIN
```

Back in 1973 a loan amortization table book had interest rates from 6% to 10%. Obviously, times have changed—oh, to find such interest rates today! About the only use this book has now is to check the accuracy of a loan amortization program.

MORTGAGE will give you the monthly payment amount needed to amortize a loan over any number of months at any desired interest rate. In line 170 the principal amount of the loan is input, and lines 190-210 obtain the yearly interest rate and the number of months for the loan to be paid off. The monthly interest rate is computed as the yearly rate divided by 12 in line 220. As you will notice, line 200 has divided the interest rate entered (say 12, for example) by 100, giving .12 for calculations. The payment amount is then derived from the formula in line 230, and lines 240-300 round the payment off to the nearest whole cent.

At this time you are asked whether you want to see a detailed schedule of payments with the declining loan balance. If you just want to compare the difference in payments between various interest rates and/or payoff periods, you may not want to see the detailed payoff schedule. However, if you want to obtain an idea of the amount of interest paid each year for tax purposes, or if you would like to know how much equity you will have in your home at any time in the future, you may want to see those exact payments and balances.

```
460 PRINT #1:"YEARLY INT: ";IY
470 PRINT #1:"PAYMENT AMT: ";PAYAMT
480 PRINT #1
490 PRINT #1:"PRINCIPLE";TAB(15);"INTEREST";
    TAB(30);
500 PRINT #1:"TO PRINCIPLE";TAB(45);
    "PAY AMT"
510 PRINT #1
520 CTRP=0
530 FOR I=1 TO N
540 INT1=PRIN*IM
550 INT2=INT1*100
560 INT3=INT(INT2)+.5
570 IF INT2>=INT3 THEN 600
580 INT2=INT(INT2)
590 GOTO 610
600 INT2=INT(INT2)+1
610 INT1=INT2/100
620 IF I<N THEN 650
640 PAYAMT=PRIN+INT1
650 TOPRIN=PAYAMT-INT1
660 PRIN=PRIN-TOPRIN
670 IF PRIN>=0 THEN 720
680 PRIN=PRIN+TOPRIN
690 PAYAMT=PRIN
700 TOPRIN=PAYAMT-INT1
710 PRIN=0
720 PRINT "P=";PRIN;TAB(16);"INT=";INT1
730 PRINT "TO P=";TOPRIN;TAB(16);"PAY=";
    PAYAMT
740 PRINT "***************PAYMENT#";I
750 CTRP=CTRP+1
760 IF PYN$="Y" THEN 810
770 IF CTRP<7 THEN 800
780 INPUT "ENTER TO CONTINUE ":X$
790 CTRP=0
800 IF PYN$<>"Y" THEN 830
810 PRINT #1:PRIN;TAB(15);INT1;TAB(30);TOPRIN;
820 PRINT #1:TAB(45);PAYAMT;TAB(55);I
```

(Continued in next column)

Furthermore, you can see the schedule of payments and balances on the screen with the display stopping when each screen is filled. (You hit **ENTER** to continue to the next screen.) Or you can obtain a printout of the same information if you have a printer. When the schedule is completed, you can then continue with another schedule or end the program. (The program is terminated by entering **0** to the principal amount prompt in line 170.)

(Continued from previous column)

```
830 NEXT I
840 IF PYN$<>"Y" THEN 860
850 CLOSE #1
860 INPUT "CONTINUE (Y/N): ":YN$
870 IF YN$="Y" THEN 160
880 END
```

```
PRINCIPAL: 5000
YEARLY INT: .12
PAYMENT AMT: 166.07

PRINCIPAL       INTEREST        TO PRINCIPAL    PAY AMT

4883.93         50              116.07          166.07          1
4766.7          48.84           117.23          166.07          2
4648.3          47.67           118.4           166.07          3
4528.71         46.48           119.59          166.07          4
4407.93         45.29           120.78          166.07          5
4285.94         44.08           121.99          166.07          6
4162.73         42.86           123.21          166.07          7
4038.29         41.63           124.44          166.07          8
3912.6          40.38           125.69          166.07          9
3785.66         39.13           126.94          166.07          10
3657.45         37.86           128.21          166.07          11
3527.95         36.57           129.5           166.07          12
3397.16         35.28           130.79          166.07          13
3265.06         33.97           132.1           166.07          14
3131.64         32.65           133.42          166.07          15
2996.89         31.32           134.75          166.07          16
2860.79         29.97           136.1           166.07          17
2723.33         28.61           137.46          166.07          18
2584.49         27.23           138.84          166.07          19
2444.26         25.84           140.23          166.07          20
2302.63         24.44           141.63          166.07          21
2159.59         23.03           143.04          166.07          22
2015.12         21.6            144.47          166.07          23
1869.2          20.15           145.92          166.07          24
1721.82         18.69           147.38          166.07          25
1572.97         17.22           148.85          166.07          26
1422.63         15.73           150.34          166.07          27
1270.79         14.23           151.84          166.07          28
1117.43         12.71           153.36          166.07          29
 962.53         11.17           154.9           166.07          30
 806.09         9.63            156.44          166.07          31
 648.08         8.06            158.01          166.07          32
 488.49         6.48            159.59          166.07          33
 327.3          4.88            161.19          166.07          34
 164.5          3.27            162.8           166.07          35
   0            1.65            164.5           166.15          36
```

SAVE1

```
100 REM **************************
110 REM *          SAVE1           *
120 REM *                          *
130 REM *     COPYRIGHT 1983       *
140 REM *   DONALD C. KREUTNER     *
150 REM **************************
160 REM
170 REM
180 PRTR$="N"
190 CALL CLEAR
200 PRINT "    $$$$$$$$$$$$$$$$"
210 PRINT "    SAVINGS ANALYSIS"
220 PRINT "    $$$$$$$$$$$$$$$$"
230 PRINT
240 PRINT
250 PRINT "IM   INVSTMT COMPOUNDED MO"
260 PRINT "IY   INVSTMT COMPOUNDED YR"
270 PRINT "MAO MO ANNUITY (ORDINARY)"
280 PRINT "YAO YR ANNUITY (ORDINARY)"
290 PRINT "MAD MO ANNUITY DUE"
300 PRINT "YAD YR ANNUITY DUE"
310 PRINT "ON   TURN PRTR ON"
320 PRINT "OFF  TURN PRTR OFF"
330 PRINT "END  END PROGRAM"
340 PRINT
350 PRINT
360 PRINT
370 REM ORDINARY ANNUITIES BEAR
380 REM NO INTEREST ON THE LAST
390 REM PAY BECAUSE THE PAYMTS
400 REM ARE DUE AT THE END OF
410 REM THE PERIOD
420 INPUT "      SELECTION: ":S$
430 IF S$="END" THEN 2540
440 IF S$="IM" THEN 640
450 IF S$="IY" THEN 1080
460 IF S$="MAO" THEN 1460
```

SAVE1 allows you to analyze various savings and investments that you can make. Basically, two major types of investments are analyzed: the investment of a lump sum, and an annuity payment that adds to a savings account on a regular basis. There are two kinds of annuities: one is the ordinary annuity, and the other is an annuity due. In an ordinary annuity, the deposits are made at the end of the period, which means that no interest is received on the last payment amount. In an annuity due, deposits are made at the beginning of each period, and interest is received on the last payment made. The difference between the two annuities is essentially a difference of timing.

The menu screen in lines 200-360 describes the possible commands that can be used:

1. IM (a lump INVESTMENT compounded MONTHLY) will analyze how much an investment of, say $500, would amount to in a certain number of months.

2. IY (a lump INVESTMENT compounded YEARLY) does the same thing over a period years.

3. MAO (a MONTHLY ANNUITY, ORDINARY type) tells you how much you would have if you had invested, say $50, every month for a period of 36 months.

```
470 IF S$="MAD" THEN 1460
480 IF S$="YAO" THEN 2030
490 IF S$="YAD" THEN 2030
500 IF S$="ON" THEN 540
510 IF S$="OFF" THEN 590
520 GOTO 190
530 REM ********************
540 IF PRTR$="Y" THEN 570
550 OPEN #1:"RS232.BA=1200"
560 PRTR$="Y"
570 GOTO 190
580 REM ********************
590 IF PRTR$="N" THEN 620
600 CLOSE #1
610 PRTR$="N"
620 GOTO 190
630 REM ********************
640 CALL CLEAR
650 PRINT "INVESTMENT COMPOUNDED MO"
660 INPUT "INVESTMENT AMOUNT: ":INV
670 INPUT "YEARLY INT RATE:    ":YI
680 IF YI>=1 THEN 710
690 PRINT "ENTER 12% AS 12"
700 GOTO 670
710 MI=YI/1200
720 INPUT "MONTHS OF INVSTMT: ":MO
730 FV=INV*(1+MI)^MO
740 FV=FV*100
750 IFV=INT(FV)
760 IF IFV+.5>FV THEN 780
770 IFV=IFV+1
780 FV=IFV/100
790 PRINT "VALUE AFTER";MO;"MOS: ";FV
800 PRINT
810 IF PRTR$<>"Y" THEN 880
820 PRINT #1
830 PRINT #1:"INVESTMENT COMPOUNDED
    MONTHLY"
840 PRINT #1:"INVESTMENT AMOUNT: ";INV
850 PRINT #1:"YEARLY INT RATE: ";YI
```

4. YAO (a YEARLY ANNUITY, ORDINARY type) analyzes an annuity payment made yearly for a certain number of years.
5. MAD (MONTHLY ANNUITY DUE) does the same as MAO, but the payments are made at the beginning of every interest period, so more interest is earned.
6. YAD (YEARLY ANNUITY DUE) is similar to YAO, but again the payments are made at the beginning of an interest period.

Each of the six different commands will ask you for a dollar amount of investment or payment (depending on whether it is for an investment or annuity). In addition, an interest rate must be keyed. For example, a yearly interest rate of 12.75% would be entered as **12.75**, and 13% would be **13**. The value after the time period requested is then displayed. If you want a detailed breakdown, you can obtain a month-by-month (or year-by-year) summary.

Specific commands of note include lines 1150-1210, which are a simple investment computation, based on the financial formula: future value = principal times (1 + the interest rate) to the number-of-payments power. The future value is then rounded off to the nearest cent in lines 1160-1200. The annuity formulas are in lines 1820-1940 and lines 2150-2220.

SAVE1, like many other programs in this book, can utilize a printer. If you have one, you can print out analyses of savings and annuity schedules. This is accomplished by

```
 860 PRINT #1:"VALUE AFTER";MO;"MOS: ";FV
 870 PRINT #1
 880 INPUT "PRINT MONTHLY DETAILS: ":YN$
 890 IF YN$<>"Y" THEN 1030
 900 FOR I=1 TO MO
 910 FV=INV*(1+MI)
 920 FVX=FV
 930 FV=FV*100
 940 IFV=INT(FV)
 950 IF IFV+.5>FV THEN 970
 960 IFV=IFV+1
 970 FV=IFV/100
 980 PRINT "MONTH ";I;TAB(18);FV
 990 IF PRTR$<>"Y" THEN 1010
1000 PRINT #1:"MONTH ";I;TAB(18);FV
1010 INV=FVX
1020 NEXT I
1030 PRINT
1040 INPUT "ENTER TO CONTINUE ":X$
1050 GOTO 190
1060 REM ******************
1070 CALL CLEAR
1080 INPUT "INVESTMENT AMOUNT: ":INV
1090 INPUT "YEARLY INT RATE:    ":YI
1100 IF YI>=1 THEN 1130
1110 PRINT "ENTER 12% AS 12"
1120 GOTO 1090
1130 YI=YI/100
1140 INPUT "YEARS OF INVSTMT:   ":YR
1150 FV=INV*(1+YI)^YR
1160 FV=FV*100
1170 IFV=INT(FV)
1180 IF IFV+.5>FV THEN 1200
1190 IFV=IFV+1
1200 FV=IFV/100
1210 PRINT "VALUE AFTER";YR;"YRS: ";FV
1220 IF PRTR$<>"Y" THEN 1270
1230 PRINT #1:"INVESTMENT AMOUNT: ";INV
1240 PRINT #1:"YEARLY INT RATE:   ";YI
1250 PRINT #1:"VALUE AFTER";YR;"YRS: ";FV
```

keying the commands **ON** or **OFF**, as described in the menu lines.

```
        $$$$$$$$$$$$$$$$$
        SAVINGS ANALYSIS
        $$$$$$$$$$$$$$$$$

IM      INVSTMT COMPOUNDED MO
IY      INVSTMT COMPOUNDED YR
MAO     MO ANNUITY (ORDINARY)
YAO     YR ANNUITY (ORDINARY)
MAD     MO ANNUITY DUE
YAD     YR ANNUITY DUE
ON      TURN PRTR ON
OFF     TURN PRTR OFF
END     END PROGRAM

        SELECTION:
```

```
YEARLY ANNUITY DUE
YEARLY ANNUITY AMT: 1000
YEARLY INT RATE:    .1
ANNUITY AMOUNT:     108181.77

YEAR   1          1100
YEAR   2          2310
YEAR   3          3641
YEAR   4          5105.1
YEAR   5          6715.61
YEAR   6          8487.17
YEAR   7          10435.89
YEAR   8          12579.48
YEAR   9          14937.42
YEAR   10         17531.17
YEAR   11         20384.28
YEAR   12         23522.71
YEAR   13         26974.98
YEAR   14         30772.48
YEAR   15         34949.73
YEAR   16         39544.7
YEAR   17         44599.17
YEAR   18         50159.09
YEAR   19         56275
YEAR   20         63002.5
YEAR   21         70402.75
YEAR   22         78543.02
YEAR   23         87497.33
YEAR   24         97347.06
YEAR   25         108181.77
```

```
1260 PRINT #1
1270 INPUT "PRINT YEARLY DETAILS: ":YN$
1280 IF YN$<>"Y" THEN 1420
1290 FOR I=1 TO YR
1300 FV=INV*(1+YI)
1310 FVX=FV
1320 FV=FV*100
1330 IFV=INT(FV)
1340 IF IFV+.5>FV THEN 1360
1350 IFV=IFV+1
1360 FV=IFV/100
1370 PRINT "YEAR ";I;TAB(18);FV
1380 IF PRTR$<>"Y" THEN 1400
1390 PRINT #1:"YEAR ";I;TAB(18);FV
1400 INV=FVX
1410 NEXT I
1420 PRINT
1430 INPUT "ENTER TO CONTINUE ":X$
1440 GOTO 190
1450 REM ********************
1460 CALL CLEAR
1470 IF S$="MAO" THEN 1500
1480 PRINT "MONTHLY ANNUITY DUE"
1490 GOTO 1510
1500 PRINT "MONTHLY ORDINARY ANNUITY"
1510 INPUT "MONTHLY ANNUITY AMT: ":PAY
1520 INPUT "YEARLY INT RATE:    ":YI
1530 IF YI>=1 THEN 1560
1540 PRINT "ENTER 12% AS 12"
1550 GOTO 1520
1560 MI=YI/1200
1570 INPUT "MONTHS OF PAYS:       ":MO
1580 AMT=(PAY/MI)*((1+MI)^MO-1)
1590 IF S$="MAO" THEN 1610
1600 AMT=AMT*(1+MI)
1610 AMT=AMT*100
1620 IAMT=INT(AMT)
1630 IF IAMT+.5>AMT THEN 1650
1640 IAMT=IAMT+1
1650 AMT=IAMT/100
```

```
INVESTMENT COMPOUNDED MONTHLY
INVESTMENT AMOUNT:  1000
YEARLY INT RATE:    13
VALUE AFTER 24 MOS: 1295.12

MONTH  1          1010.83
MONTH  2          1021.78
MONTH  3          1032.85
MONTH  4          1044.04
MONTH  5          1055.35
MONTH  6          1066.79
MONTH  7          1078.34
MONTH  8          1090.02
MONTH  9          1101.83
MONTH  10         1113.77
MONTH  11         1125.84
MONTH  12         1138.03
MONTH  13         1150.36
MONTH  14         1162.82
MONTH  15         1175.42
MONTH  16         1188.15
MONTH  17         1201.03
MONTH  18         1214.04
MONTH  19         1227.19
MONTH  20         1240.48
MONTH  21         1253.92
MONTH  22         1267.51
MONTH  23         1281.24
MONTH  24         1295.12
```

```
MONTHLY ORDINARY ANNUITY
MONTHLY ANNUITY AMT: 200
YEARLY INT RATE:     13
ANNUITY AMOUNT:      2548.29

MONTH  1          200
MONTH  2          402.17
MONTH  3          606.52
MONTH  4          813.09
MONTH  5          1021.9
MONTH  6          1232.97
MONTH  7          1446.33
MONTH  8          1662
MONTH  9          1880
MONTH  10         2100.37
MONTH  11         2323.12
MONTH  12         2548.29
```

```
1660 PRINT "ANNUITY AMOUNT: ";AMT
1670 PRINT
1680 IF PRTR$<>"Y" THEN 1770
1690 IF S$="MAO" THEN 1720
1700 PRINT #1:"MONTHLY ANNUITY DUE"
1710 GOTO 1730
1720 PRINT #1:"MONTHLY ORDINARY ANNUITY"
1730 PRINT #1:"MONTHLY ANNUITY AMT: ";PAY
1740 PRINT #1:"YEARLY INT RATE:     ";YI
1750 PRINT #1:"ANNUITY AMOUNT:      ";AMT
1760 PRINT #1
1770 INPUT "PRINT MONTHLY DETAILS: ":YN$
1780 IF YN$<>"Y" THEN 1990
1790 AMT=0
1800 FOR I=1 TO MO
1810 IF S$="MAO" THEN 1830
1820 AMT=AMT+PAY
1830 FV=AMT*(1+MI)
1840 FVX=FV
1850 FV=FV*100
1860 IFV=INT(FV)
1870 IF IFV+.5>FV THEN 1890
1880 IFV=IFV+1
1890 FV=IFV/100
1900 IF S$="MAD" THEN 1940
1910 FV=FV+PAY
1920 AMT=FVX+PAY
1930 GOTO 1950
1940 AMT=FVX
1950 PRINT "MONTH ";I;TAB(18);FV
1960 IF PRTR$<>"Y" THEN 1980
1970 PRINT #1:"MONTH ";I;TAB(18);FV
1980 NEXT I
1990 PRINT
2000 INPUT "ENTER TO CONTINUE ":X$
2010 GOTO 190
2020 REM ********************
2030 CALL CLEAR
2040 IF S$="YAO" THEN 2070
```

(Continued in next column)

```
INVESTMENT AMOUNT:  1500
YEARLY INT RATE:    .128
VALUE AFTER 10 YRS: 5002.44

YEAR   1          1692
YEAR   2          1908.58
YEAR   3          2152.87
YEAR   4          2428.44
YEAR   5          2739.28
YEAR   6          3089.91
YEAR   7          3485.42
YEAR   8          3931.55
YEAR   9          4434.79
YEAR   10         5002.44
```

(Continued from previous column)

```
2050 PRINT "YEARLY ANNUITY DUE"
2060 GOTO 2080
2070 PRINT "YEARLY ORDINARY ANNUITY"
2080 INPUT "YEARLY ANNUITY AMT: ":PAY
2090 INPUT "YEARLY INT RATE:    ":YI
2100 IF YI>=1 THEN 2130
2110 PRINT "ENTER 12% AS 12"
2120 GOTO 2090
2130 YI=YI/100
2140 INPUT "YEARS OF PAYS:      ":YR
2150 AMT=(PAY/YI)*((1+YI)^YR-1)
2160 IF S$="YAO" THEN 2180
2170 AMT=AMT*(1+YI)
2180 AMT=AMT*100
2190 IAMT=INT(AMT)
2200 IF IAMT+.5>AMT THEN 2220
2210 IAMT=IAMT+1
2220 AMT=IAMT/100
2230 PRINT "ANNUITY AMOUNT: ";AMT
2240 PRINT
2250 IF PRTR$<>"Y" THEN 2340
```

```
2260 IF S$="YAO" THEN 2290
2270 PRINT #1:"YEARLY ANNUITY DUE"
2280 GOTO 2300
2290 PRINT #1:"YEARLY ORDINARY ANNUITY"
2300 PRINT #1:"YEARLY ANNUITY AMT: ";PAY
2310 PRINT #1:"YEARLY INT RATE:    ";YI
2320 PRINT #1:"ANNUITY AMOUNT:     ";AMT
2330 PRINT #1
2340 INPUT "PRINT YEARLY DETAILS: ":YN$
2350 IF YN$<>"Y" THEN 2510
2360 AMT=0
2370 FOR I=1 TO YR
2372 IF S$="YAO" THEN 2380
2375 AMT=AMT+PAY
2380 FV=AMT*(1+YI)
2390 FVX=FV
2400 FV=FV*100
2410 IFV=INT(FV)
2420 IF IFV+.5>FV THEN 2440
2430 IFV=IFV+1
2440 FV=IFV/100
2445 IF S$="YAD" THEN 2465
2450 FV=FV+PAY
2460 AMT=FVX+PAY
2462 GOTO 2470
2465 AMT=FVX
2470 PRINT "YEAR ";I;TAB(18);FV
2480 IF PRTR$<>"Y" THEN 2500
2490 PRINT #1:"YEAR   ";I;TAB(18);FV
2500 NEXT I
2510 PRINT
2520 INPUT "ENTER TO CONTINUE ":X$
2530 GOTO 190
2540 END
```

SCORE1

```
100 REM * ************************
110 REM *                        *
120 REM *        SCORE1          *
130 REM *     COPYRIGHT 1983     *
140 REM *                        *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 REM SCORE1
180 OPTION BASE 1
190 DIM N(50)
200 DIM N$(50)
210 DIM SCORELAST(50)
220 DIM SCORENEW(50)
230 DIM NEWTOT(50)
240 FOR I=1 TO 50
250 N$(I)="@"
260 SCORELAST(I)=0
270 SCORENEW(I)=0
280 N(I)=I
290 NEWTOT(I)=0
300 NEXT I
310 DATA "01 DIRK & SUSAN MULLINGER"
320 DATA "02 DON & JANET KREUTNER"
330 DATA "03 RON & MARY BROWN"
340 DATA "04 JACK & CAROL MASON"
350 DATA "05 SCOTT & SHARON MCALLISTER"
360 DATA "06 ROGER & BARBARA HOWARD"
370 DATA "07 BOB & SALLY BARD"
380 DATA "08 TONY & HARRIET HUGHES"
390 DATA "END"
400 CTR=0
410 FOR I=1 TO 50
420 READ NAME$
430 IF NAME$="END" THEN 470
440 CTR=CTR+1
450 N$(I)=NAME$
460 NEXT I
```

SCORE1 can perform a variety of score-keeping applications, ranging from keeping track of a bridge club's monthly and cumulative scores, to accumulating a team's scores in various games, to keeping a rudimentary general-ledger total of money spent on certain categories of expenses and money received from certain sources.

Obviously, the program name implies that it was designed for scorekeeping purposes, but it can keep track of many different totals. The categories used by the program are defined in the **DATA** statements in lines 310-390. You are not limited to this number of items, since you can have up to 50 classes with which to "score" totals. You can have several copies of this program with different **DATA** statements. Then you can save various team totals on tape or disk and later add to them either regularly or irregularly.

The following five arrays are used to accumulate the data:

1. The number (must be 1 to 50) of the team or group (N)
2. The name of the category or group (N$)
3. The last score or total of the category (SCORELAST)
4. The new score to be added to the SCORELAST (SCORENEW)

```
470 FOR I=1 TO CTR
480 PRINT "ENTRY # ";I
490 PRINT N$(I)
500 INPUT "LAST TOTAL: ":SCORELAST(I)
510 INPUT "NEW SCORE: ":SCORENEW(I)
520 NEWTOT(I)=SCORELAST(I)+SCORENEW(I)
530 NEXT I
540 REM ************************
550 INPUT "TITLE OF REPORT: ":T$
560 INPUT "PRINTER (Y/N): ":PYN$
570 IF PYN$<>"Y" THEN 590
580 OPEN #1:"RS232.BA=1200"
590 T2$="NUMERICAL SEQUENCE"
600 FLAG$="N"
610 FOR I=1 TO CTR-1
620 IF N(I)<=N(I+1)THEN 640
630 GOSUB 880
640 NEXT I
650 IF FLAG$="Y" THEN 600
660 GOSUB 1050
670 FLAG$="N"
680 FOR I=1 TO CTR-1
690 IF NEWTOT(I)>=NEWTOT(I+1)THEN 710
700 GOSUB 880
710 NEXT I
720 IF FLAG$="Y" THEN 670
730 T2$="NEW TOTAL SEQUENCE"
740 GOSUB 1050
750 FLAG$="N"
760 FOR I=1 TO CTR-1
770 IF SCORENEW(I)>=SCORENEW(I+1)THEN 790
780 GOSUB 880
790 NEXT I
800 IF FLAG$="Y" THEN 750
810 T2$="CURRENT SCORE SEQUENCE"
820 GOSUB 1050
830 INPUT "REPEAT (Y/N): ":RYN$
840 IF RYN$="Y" THEN 590
850 IF PYN$<>"Y" THEN 870
860 CLOSE #1
```

5. The new total or accumulation of SCORELAST and SCORENEW (NEWTOT)

The **DIM** statements in lines 190-230 set up these arrays, and the loop in lines 240-300 initializes the numeric values to zero and the names to "@" for all names. Line 400 sets the counter CTR to zero. This variable keeps track of how many items are entered, so that the sorts and printouts that follow "know" how many items are to be evaluated. (Remember that up to 50 are possible as the program has been set up.) If you have more than 50 items, you can enter the data in two or more groups, using programs with different **DATA** statements. In the lines 410-530, you enter the applicable input information. You must key a previous score and the new score for each as you are prompted (even if both are zero). The program will ask for an old and new total for each number from 1 to CTR (the total number of **DATA** statements numbered from 01 to 50, or as high a number as you want).

Line 550 asks for the name of the report, which you key in (no commas, unless you enclose the title in quotation marks). Then you are asked if you have a printer. (If you do not, the report will be displayed on the screen.) Three orders will be printed: numerical sequence by number (01 through 50), descending order by total scores, and descending order by the current score only. Each report is preceded by a bubble sort (see the description in the program BSKTSTAT) to arrange the data in the proper sequence for the desired report. Each report is performed by the same subroutine

```
870 GOTO 1390
880 NEWTOTX=NEWTOT(I)
890 NEWTOT(I)=NEWTOT(I+1)
900 NEWTOT(I+1)=NEWTOTX
910 NX=N(I)
920 N(I)=N(I+1)
930 N(I+1)=NX
940 NX$=N$(I)
950 N$(I)=N$(I+1)
960 N$(I+1)=NX$
970 SCORELASTX=SCORELAST(I)
980 SCORELAST(I)=SCORELAST(I+1)
990 SCORELAST(I+1)=SCORELASTX
1000 SCORENEWX=SCORENEW(I)
1010 SCORENEW(I)=SCORENEW(I+1)
1020 SCORENEW(I+1)=SCORENEWX
1030 FLAG$="Y"
1040 RETURN
1050 INPUT "HIT ENTER TO CONTINUE ":X$
1060 CTR2=0
1070 CALL CLEAR
1080 PRINT T$
1090 PRINT T2$
1100 PRINT
1110 IF PYN$<>"Y" THEN 1220
1120 PRINT #1:"****************************";
1130 PRINT #1:"****************************";
1140 PRINT #1:"************************"
1150 PRINT #1
1160 PRINT #1:T$
1170 PRINT #1:T2$
1180 PRINT #1
1190 PRINT #1:TAB(40);"OLD TOT";TAB(52);
     "NEW TOT";
1200 PRINT #1:TAB(64);"CURRENT"
1210 PRINT #1
1220 FOR I=1 TO CTR
1230 CTR2=CTR2+1
1240 IF CTR2<5 THEN 1270
```

(Continued in next column)

GOSUB 1050, which prints a different subtitle based on the value of T2$ (assigned in lines 590, 730, and 810).

If the report is displayed on the screen only, the scrolling is stopped by keeping track of how many totals have been printed (CTR2 in lines 1230-1260). In this manner, when a screen fills up, the program will wait for you to hit **ENTER** before it rolls the previous information off the screen. Furthermore, you can repeat the report again and again by answering **Y** to the input statement in line 830.

Notice that in the sort routines, the same subroutine, **GOSUB** 880, is used to switch elements if they are out of order.

(Continued from previous column)

```
1250 CTR2=1
1260 INPUT "ENTER TO CONTINUE ":X$
1270 PRINT I;N$(I)
1280 PRINT "OLD TOTAL: ";SCORELAST(I)
1290 PRINT "NEW TOTAL: ";NEWTOT(I)
1300 PRINT "CURRENT:    ";SCORENEW(I)
1310 PRINT "****************************"
1320 IF PYN$<>"Y" THEN 1360
1330 PRINT #1:I;"- ";N$(I);TAB(40);SCORELAST(I);
1340 PRINT #1:TAB(52);NEWTOT(I);TAB(64);
1350 PRINT #1:SCORENEW(I)
1360 NEXT I
1365 IF PYN$<>"Y" THEN 1380
1370 PRINT #1
1380 RETURN
1390 END
```

```
**********************************************************************************

SAMPLE ACCUMULATED SCORES
NUMERICAL SEQUENCE

                                    OLD TOT       NEW TOT       CURRENT

 1 - 01 DIRK & SUSAN MULLENGER        21124         22222         1098
 2 - 02 DON & JANET KREUTNER          20013         21012          999
 3 - 03 RON & MARY BROWN              18900         19605          705
 4 - 04 JACK & CAROL MASON            20996         22196         1200
 5 - 05 SCOTT & SHARON MCALLISTER     18002         19025         1023
 6 - 06 ROGER & BARBARA HOWARD        19072         20195         1123
 7 - 07 BOB & SALLY BARD              19201         20501         1300
 8 - 08 TONY & HARRIET HUGHES         20100         20997          897

**********************************************************************************

SAMPLE ACCUMULATED SCORES
NEW TOTAL SEQUENCE

                                    OLD TOT       NEW TOT       CURRENT

 1 - 01 DIRK & SUSAN MULLENGER        21124         22222         1098
 2 - 04 JACK & CAROL MASON            20996         22196         1200
 3 - 02 DON & JANET KREUTNER          20013         21012          999
 4 - 08 TONY & HARRIET HUGHES         20100         20997          897
 5 - 07 BOB & SALLY BARD              19201         20501         1300
 6 - 06 ROGER & BARBARA HOWARD        19072         20195         1123
 7 - 03 RON & MARY BROWN              18900         19605          705
 8 - 05 SCOTT & SHARON MCALLISTER     18002         19025         1023

**********************************************************************************

SAMPLE ACCUMULATED SCORES
CURRENT SCORE SEQUENCE

                                    OLD TOT       NEW TOT       CURRENT

 1 - 07 BOB & SALLY BARD              19201         20501         1300
 2 - 04 JACK & CAROL MASON            20996         22196         1200
 3 - 06 ROGER & BARBARA HOWARD        19072         20195         1123
 4 - 01 DIRK & SUSAN MULLENGER        21124         22222         1098
 5 - 05 SCOTT & SHARON MCALLISTER     18002         19025         1023
 6 - 02 DON & JANET KREUTNER          20013         21012          999
 7 - 08 TONY & HARRIET HUGHES         20100         20997          897
 8 - 03 RON & MARY BROWN              18900         19605          705
```

2

Games and Miscellaneous Programs

BEEP1

```
100 REM * ************************ *
110 REM *                          *
120 REM *          BEEP1           *
130 REM *                          *
140 REM *      COPYRIGHT 1983      *
150 REM *    DONALD C. KREUTNER    *
160 REM * ************************ *
170 REM ***PROGRAM BEEP1***
180 CALL CLEAR
190 FOR I=1 TO 20
200 CALL SOUND(100,600,15)
210 NEXT I
220 CALL CLEAR
230 INPUT "WHAT TO SAY? (XX=QUIT) ":X$
240 IF X$="SAME" THEN 330
250 IF X$="XX" THEN 410
260 L$=X$
270 INPUT "HOW MANY BEEPS? ":B
280 INPUT "CLEAR SCREEN (Y/N): ":YN$
290 INPUT "DURATION (1 TO 4250): ":D
300 INPUT "PITCH (110 TO 44733): ":P
310 INPUT "LOUD OR SOFT (0 LOUDEST -
    30)? ":L
```

BEEP1 is a "fun" program that allows you to key any message you want repeated over and over on the screen. At the same time the message is displayed, you can have a beep or tone sounded at any pitch, at any level of loudness, and for any duration.

When the message has been selected in line 230, you are then prompted for the number of beeps in line 270. If you want the display to last a long time, just enter a high number. The screen is cleared after each beep if you select the clear-screen feature (line 280). And if you want to repeat the same message on completion, you need to key only **SAME** to the "WHAT TO SAY?" prompt. If you do, you are not asked for information about duration, pitch, and loudness.

Note that at program startup and before the "WHAT TO SAY?" prompt, a loop of 20 tones is sounded to catch your attention.

```
320 GOTO 340
330 X$=L$
340 FOR I=1 TO B
350 IF YN$<>"Y" THEN 370
360 CALL CLEAR
370 PRINT I;TAB(6);X$
380 CALL SOUND(D,P,L)
390 NEXT I
400 GOTO 180
410 END
```

BIRTHDAY

```
100 REM * ************************* *
110 REM *                           *
120 REM *          BIRTHDAY         *
130 REM *                           *
140 REM *       COPYRIGHT 1983      *
150 REM *     DONALD C. KREUTNER    *
160 REM * ************************* *
170 DATA 01/03/71 ROBERT JAY
180 DATA 08/08/45 BARD JANET
185 DATA 11/19/18 BARD SALLY
190 DATA 06/07/69 SEAN DEREK
195 DATA 12/21/19 BARD ROBERT
200 DATA 10/11/37 DEXTER EARL
210 DATA 03/14/59 SANDLIN DONNA
220 DATA 01/23/63 BEYL ANDREW
230 DATA 12/23/58 STIDHAM DEBRA
240 DATA 09/28/57 FORD JUDY
250 DATA 06/29/59 MEYER DIANE
260 DATA 08/03/42 ANN CAROL
270 DATA 12/03/56 WHITSITT DEBBIE
280 DATA 06/25/45 CHARLES DON
290 DATA 03/19/55 MAY SUSAN
300 DATA 03/15/15 JOHN ALBERT
310 DATA 01/26/19 KREUTNER PAULINE
320 DATA 12/29/56 WYNE GREG
330 DATA 11/11/36 LONGEST NORMAN
340 DATA 05/19/21 MCKNIGHT MARTHA
350 DATA 02/01/49 JOHNSON BARBARA
351 DATA 02/25/71 MASON KEVIN
352 DATA 08/29/78 MASON ANDREA
353 DATA 04/28/44 MASON JACK
354 DATA 04/15/71 HEALY CHRIS
355 DATA 08/18/73 HEALY KARA
356 DATA 03/23/69 HEALY MARJAN
360 DATA 07/05/53 WHITE JAMES
370 DATA 10/31/58 WEST KEVIN
380 DATA 02/19/41 CAPITO BETTY
```

BIRTHDAY is a very useful program that can (1) tell you all your acquaintances who have birthdays in any given month, or (2) provide you with an alphabetical list of people with their birthdays.

This program also can be used for many other purposes. For example, it can give you a calendar of events for any given month or let you keep an alphabetical list of event names. On January 10th, for example, you may have ten things listed for you to remember, whereas on the 11th, you may have three. If you want to see all the appointments you have with a certain firm, you can obtain a sorted list (provided, of course, that you were careful to enter the "names" of those appointments in the same way each time).

BIRTHDAY can be used, then, in a variety of ways to keep a calendar of events so that you don't have to keep track of all of them in your head. With BIRTHDAY you won't forget an anniversary, a birthday, or an important appointment.

The **DATA** statements in lines 170 to 440 can be replaced by your own statements, and you can have multiple copies of this program to serve multiple purposes (a birthday version, an anniversary version, a personal calendar version, or a combination of these). To do so, just key your own **DATA** statements instead of the ones in the listing (unless you want to remember my birthday list). Note that a final **DATA** statement of "99/99/99 ZZZZZ" is needed to indicate to

```
390 DATA 10/26/70 HOWARD TODD
400 DATA 05/15/70 THOM BILLY
410 DATA 02/23/71 GOLDMAN JOHN
420 DATA 12/26/70 STRIEGEL JOHN
430 DATA 11/06/67 ASBERRY SHAWN
440 DATA 99/99/99 ZZZZZ
450 DIM X$(200)
460 FOR I=1 TO 200
470 READ X$(I)
480 IF SEG$(X$(I),1,2)="99" THEN 500
490 NEXT I
500 CTR=I
510 FOR J=1 TO I
520 PRINT X$(J)
530 NEXT J
540 CALL CLEAR
550 PRINT "##   BIRTHDAYS IN MONTH ##"
560 PRINT "    (13=ALL MONTHS)"
570 PRINT "A   BIRTHDAYS (ALP ORDER)"
580 PRINT "END  END PROGRAM"
590 PRINT
600 INPUT "SELECTION: ":S$
610 IF S$="A" THEN 1030
620 IF S$="END" THEN 1350
630 S1$=SEG$(S$,1,1)
640 S2$=SEG$(S$,2,1)
650 IF S1$<"0" THEN 540
660 IF S1$>"9" THEN 540
670 IF S2$<"0" THEN 540
680 IF S2$>"9" THEN 540
690 M$=S1$&S2$
700 PASS=0
710 IF LEN(M$)<>2 THEN 540
720 SW$="N"
730 PASS=PASS+1
740 CALL CLEAR
750 PRINT "SORT PASS";PASS
760 FOR I=1 TO CTR-1
770 IF SEG$(X$(I),1,5)<=SEG$(X$(I+1),1,5)
    THEN 820
```

the data **READ** statement of line 470 that all of the input data has been read. You can add or delete input data by adding or taking away **DATA** statements.

Now let's look at some of the program statements that are of interest. After the **DATA** statements have been read into the X$ array in lines 460-490, your choices are as follows:

1. List all the birthdays or appointments in any month number (01 through 12).
2. List in alphabetical order all names or appointment descriptions.
3. End the program.

These choices are given in lines 540-600.

If you key **A** (for an alphabetical list), control passes to line 1030 from line 610. **END** causes the program to end, going to statement 1350 from line 620. Any other selection is analyzed by lines 610-710 to make certain a two-digit number was keyed. This check is accomplished by looking at S1$ (the first character) and S2$ (the second character) to make sure they are between 0 and 9. The check is not quite perfect, since months higher than 12 could be requested. The result of such a request, however, would not be disastrous.

If a month number is keyed, the next step that the program accomplishes is a sort of the data to rearrange it in order of the first 5 characters (the month, a slash, and the day). This sorting is done by a bubble sort in lines 720-830. Note that each time you request a month's appointments (or other

```
780 Y$=X$(I)
790 X$(I)=X$(I+1)
800 X$(I+1)=Y$
810 SW$="Y"
820 NEXT I
830 IF SW$="Y" THEN 720
840 CTRX=0
850 INPUT "PRINTER (Y/N): ":PYN$
860 IF PYN$<>"Y" THEN 880
870 OPEN #1:"RS232.BA=1200"
880 FOR I=1 TO CTR
890 IF M$="13" THEN 910
900 IF SEG$(X$(I),1,2)<>M$ THEN 980
910 PRINT X$(I)
920 IF PYN$<>"Y" THEN 940
930 PRINT #1:X$(I)
940 CTRX=CTRX+1
950 IF CTRX<20 THEN 980
960 CTRX=0
970 INPUT "ENTER TO CONTINUE: ":E$
980 NEXT I
990 IF PYN$<>"Y" THEN 1010
1000 CLOSE #1
1010 INPUT "ENTER TO CONTINUE: ":E$
1020 GOTO 540
1030 PASS=0
1040 SW$="N"
1050 PASS=PASS+1
1060 CALL CLEAR
1070 PRINT "SORT PASS";PASS
1080 FOR I=1 TO CTR-1
1090 S1$=SEG$(X$(I),10,LEN(X$(I))-9)
1100 S2$=SEG$(X$(I+1),10,LEN(X$(I+1))-9)
1110 IF S1$<=S2$ THEN 1160
1120 SW$="Y"
1130 Y$=X$(I)
1140 X$(I)=X$(I+1)
1150 X$(I+1)=Y$
1160 NEXT I
1170 IF SW$="Y" THEN 1040
```

information), you will "re-sort" the data. You may think this process is rather inefficient. However, if the data is already in month/day order, the sort pass checks only one time through the data to verify that the elements are already in sequence. Remember from previous examples that in a bubble sort a flag is kept to see if any elements were out of order and switched. If not, the sort is terminated. The flag is SW$ in lines 720, 810, and 830.

Next, you are asked if you want a listing on a printer or only on the screen. (See lines 850-870.) Only the data from the month selected will be displayed because the first two characters must be equal to the month selected. (See line 900.) Notice also that if you key **13** as the month selected, all data is displayed in month/day sequence. Now you can see that if you key a number higher than 13, the data is sorted (or re-sorted) in month/day sequence, but no detail data is displayed—unless, of course, you actually chose to key into your **DATA** statements "months" higher than 12.

If you select an alphabetical list, a sort of the data in alphabetical order is accomplished by lines 1040-1170. The sort key used is from the tenth position of the data (after the space following the eight-character date) to the end of each data element. To sort a group of names, the length of each data string must be known. The reason is that lines 1090 and 1100 analyze the names by using the **SEG$** function. For example, if you tried to get a segment string by instructing the program to start in the tenth position for a length of 20 characters in every element, some elements would not

```
1180 CTRX=0
1190 INPUT "PRINTER (Y/N): ":PYN$
1200 IF PYN$<>"Y" THEN 1220
1210 OPEN #1:"RS232.BA=1200"
1220 FOR I=1 TO CTR
1230 PRINT X$(I)
1240 IF PYN$<>"Y" THEN 1260
1250 PRINT #1:X$(I)
1260 CTRX=CTRX+1
1270 IF CTRX<20 THEN 1300
1280 CTRX=0
1290 INPUT "ENTER TO CONTINUE: ":E$
1300 NEXT I
1310 IF PYN$<>"Y" THEN 1330
1320 CLOSE #1
1330 INPUT "ENTER TO CONTINUE: ":E$
1340 GOTO 540
1350 END
```

be that long, and the program would stop in an error condition. Again, a printout in alphabetical order can be obtained. If you do not have a printer, 20 items at a time are displayed on the screen. After each display, you must hit **ENTER** to continue the screen listing.

```
01/03/71 ROBERT JAY
01/23/63 BEYL ANDREW
01/26/19 KREUTNER PAULINE
02/01/49 JOHNSON BARBARA
02/19/41 CAPITO BETTY
02/23/71 GOLDMAN JOHN
02/25/71 MASON KEVIN
03/14/59 SANDLIN DONNA
03/15/15 JOHN ALBERT
03/19/55 MAY SUSAN
03/23/69 HEALY MARJAN
04/15/71 HEALY CHRIS
04/28/44 MASON JACK
05/15/70 THOM BILLY
05/19/21 MCKNIGHT MARTHA
06/07/69 SEAN DEREK
06/25/45 CHARLES DON
06/29/59 MEYER DIANE
07/05/53 WHITE JAMES
08/03/42 ANN CAROL
08/08/45 BARD JANET
08/18/73 HEALY KARA
08/29/78 MASON ANDREA
09/28/57 FORD JUDY
10/11/37 DEXTER EARL
10/26/70 HOWARD TODD
10/31/58 WEST KEVIN
11/06/67 ASBERRY SHAWN
11/11/36 LONGEST NORMAN
11/19/18 BARD SALLY
12/03/56 WHITSITT DEBBIE
12/21/19 BARD ROBERT
12/23/58 STIDHAM DEBRA
12/26/70 STRIEGEL JOHN
12/29/56 WYNE GREG
99/99/99 ZZZZZ
08/03/42 ANN CAROL
11/06/67 ASBERRY SHAWN
08/08/45 BARD JANET
12/21/19 BARD ROBERT
11/19/18 BARD SALLY
01/23/63 BEYL ANDREW
02/19/41 CAPITO BETTY
06/25/45 CHARLES DON
10/11/37 DEXTER EARL
09/28/57 FORD JUDY
02/23/71 GOLDMAN JOHN
04/15/71 HEALY CHRIS
08/18/73 HEALY KARA
03/23/69 HEALY MARJAN
10/26/70 HOWARD TODD
03/15/15 JOHN ALBERT
02/01/49 JOHNSON BARBARA
01/26/19 KREUTNER PAULINE
```

DRAW1

```
100 REM * ************************
110 REM *                        *
120 REM *         DRAW1          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 INPUT "* OR S FOR SQUARE: ":A$
180 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
190 DIM X(20,30)
200 FOR I=1 TO 20
210 FOR J=1 TO 30
220 PRINT "ROW ";I;"POSITION ";J;
230 PRINT "(0  = ADVANCE TO NEXT ROW)"
240 PRINT "(999 = DRAW THE DIAGRAM)"
250 PRINT "(998 = END PROGRAM)"
260 PRINT "(## = COL IN ROW TO USE)"
270 INPUT " ":N
280 X(I,J)=N
290 IF N=0 THEN 330
300 IF N=999 THEN 340
310 IF N=998 THEN 610
320 NEXT J
330 NEXT I
340 CALL CLEAR
350 INPUT "# OF TIMES: ":N2
360 N3=0
370 CALL CLEAR
380 FOR I=1 TO 20
390 FOR J=1 TO 30
400 IF X(I,J)=0 THEN 470
410 IF X(I,J)=999 THEN 510
420 N=X(I,J)
430 IF A$<>"S" THEN 460
440 CALL VCHAR(I+4,X(I,J),96)
450 GOTO 470
460 PRINT TAB(N);"*";
470 NEXT J
```

DRAW1 enables you to "draw" any picture or diagram you want on the screen, using block characters. If you first lay out your picture on graph paper by filling in some blocks and leaving others blank, you can then duplicate your drawing on the screen.

Your picture can be up to 20 rows deep (up and down) and 30 columns wide (across the screen). Line 170 asks you if you want to use blocks (squares) or asterisks (*'s) to draw the diagram. You must answer either **S** or *.

Then you need to tell your 99/4A what columns in each row are to be "filled in." The program will start out with ROW 1 and POSITION 1. You key the first column (a number from 1 to 30) in which you want a square. Then you are asked for ROW 1, POSITION 2, to which you key the second column where you want a square. This process is repeated for ROW 1 until you type **0** (zero) to advance to the next row, **998** to end the program, or **999** to draw the picture.

The actual drawing takes place in lines 350-570. You are asked in line 350 how many times you want to draw the picture, have it erased, and then have it redrawn. If you selected squares, the **VCHAR** call is used to put the blocks in the proper rows and columns. Or if you selected asterisks, **PRINT**s and **TAB**s to columns are used.

When the number of repetitions you requested has been displayed, you can repeat the same diagram (line 580), or you can key in a new one.

```
480 IF A$="S" THEN 500
490 PRINT
500 NEXT I
510 IF A$="S" THEN 550
520 PRINT
530 PRINT
540 PRINT
550 N3=N3+1
560 IF N3>=N2 THEN 580
570 GOTO 370
580 INPUT "REPEAT (Y/N) ":YN$
590 IF YN$="Y" THEN 340
600 GOTO 200
610 PRINT
620 PRINT
630 PRINT
640 END
```

EDIT1

```
100 REM * ************************ *
110 REM *                          *
120 REM *          EDIT1           *
130 REM *                          *
140 REM *       COPYRIGHT 1983     *
150 REM *     DONALD C. KREUTNER   *
160 REM * ************************ *
170 DIM E$(300)
180 E$(1)="END"
190 REM ***EDIT1***
200 INPUT "W/R/C/E/L/P/S (WRITE/READ/
    CHECK TAPE/EDIT/LIST/PRINT/STOP): ":
    WR$
210 IF WR$="W" THEN 290
220 IF WR$="R" THEN 450
230 IF WR$="L" THEN 760
240 IF WR$="E" THEN 950
250 IF WR$="P" THEN 1050
260 IF WR$="C" THEN 630
270 IF WR$="S" THEN 1140
280 GOTO 200
290 INPUT "CASSETTE OR DISK (C/D): ":CD$
300 IF CD$="C" THEN 350
310 IF CD$<>"D" THEN 290
320 INPUT "DSK1.FILENAME: ":D$
330 OPEN #1:D$
340 GOTO 360
350 OPEN #1:"CS1",INTERNAL,OUTPUT,
    FIXED 128
360 FOR I=1 TO 300
370 A$=E$(I)
380 PRINT I;"*** W/TAPE ***"
390 PRINT A$
400 PRINT #1:A$
410 IF A$="END" THEN 430
420 NEXT I
430 CLOSE #1
```

EDIT1 is one of two programs (EDIT2 is the other) in this book that can create a multi-purpose file to be saved on disk or tape. When you use either of these two programs, files can be read back, and lines can be deleted or added to correct a file to the condition you want. You can use these files to keep any important details or records you may later want to review.

Both programs write data in the same format so that files created by one can be modified by the other. In addition, several programs in this book have been written so that they can read data created by EDIT1 and EDIT2. You can easily create files, add to them, or modify them. In this chapter the programs that use files created by either EDIT1 or EDIT2 include LABEL1, RECIPE1, and SORT1.

In the EDIT1 program the commands available are the following: **READ** a file; **WRITE** a file; **EDIT** a file that has been **READ** in; **LIST** a file the way it has been **READ** or **EDIT**ed; **PRINT** a file; **CHECK** a tape file for accuracy; or **STOP** the program. Lines 200-270 select the appropriate actions based on the response you key in line 200.

The **READ (R)** action is implemented by lines 450-750. You are given two sub-options (to read from tape or disk). If you request disk, you are then prompted for the disk file name in line 480. Otherwise, the cassette tape is read. Any file that you input from tape or disk will replace a previous file that you had read in or entered. Each

```
440 GOTO 200
450 INPUT "CASSETTE OR DISK (C/D): ":CD$
460 IF CD$="C" THEN 510
470 IF CD$<>"D" THEN 450
480 INPUT "DSK1.FILENAME: ":D$
490 OPEN #1:D$
500 GOTO 520
510 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
520 I=0
530 INPUT #1:X$
540 A$=X$
550 I=I+1
560 E$(I)=X$
570 PRINT I;"*** R/TAPE ***"
580 PRINT A$
590 IF A$="END" THEN 610
600 GOTO 530
610 CLOSE #1
620 GOTO 200
630 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
640 I=0
650 E=0
660 INPUT #1:X$
670 I=I+1
680 IF E$(I)=X$ THEN 700
690 E=E+1
700 IF X$="END" THEN 730
710 IF I>300 THEN 730
720 GOTO 660
730 CLOSE #1
740 PRINT "ERRORS = ";E
750 GOTO 200
760 INPUT "BEG LINE#: ":B
770 INPUT "END LINE#: ":END1
780 INPUT "DELAY (200-1000): ":DELAY
790 INPUT "LIST LINE#'S? (Y/N): ":YN$
800 IF B<1 THEN 760
810 IF B>300 THEN 760
820 IF END1<B THEN 760
830 IF END1<1 THEN 760
```

record is read into the array E$, starting from position 1 (line 520 sets the subscript to 0) and continuing until a record is read containing the word "END" only, which marks the end of the file (see line 590).

You can **EDIT** (function **E**) the file in memory; or if you desire to create a new file, you can start putting new lines or records into memory, beginning at position 1 and using the **E** command. Line 950 asks you for the line # you desire to edit. If you select line number 1, you can erase all records with the same record sequence number you are keying.

It is important to note that how you terminate the edit mode can affect what lines are left intact in memory. You can key **//** to stop editing (or replacing) lines, thus leaving alone the lines following. However, be sure that some record later on has only the word "END" in it since the END record is the signal to the **WRITE** command that the end of the file has been reached. Besides using **//**, you can key a record with "END" only to terminate the edit mode. This method will erase any records following the END record, as far as the **WRITE** command is concerned. Keying **STOP** to end the edit mode has the same effect as keying **//**. Then succeeding records are left intact.

For a brief example, let's suppose you **READ** in from cassette a file containing the following records:

```
This is record number 1
And this is # 2
This is the third record
END
```

```
840 IF END1>300 THEN 760
850 FOR I=B TO END1
860 IF YN$<>"Y" THEN 880
870 PRINT I;"***"
880 PRINT E$(I)
890 IF E$(I)="END" THEN 200
900 REM ***NEXT 2 LINES FOR DELAY TO READ
    SCREEN***
910 FOR J=1 TO DELAY
920 NEXT J
930 NEXT I
940 GOTO 200
950 INPUT "LINE TO EDIT: ":E
960 IF E>300 THEN 200
970 PRINT E;"*** EDIT ***"
980 INPUT A$
990 IF A$="STOP" THEN 200
1000 IF A$="//" THEN 200
1010 E$(E)=A$
1020 IF A$="END" THEN 200
1030 E=E+1
1040 GOTO 960
1050 OPEN #2:"RS232.BA=1200"
1060 INPUT "BEG LINE#: ":B
1070 INPUT "END LINE#: ":END1
1080 FOR I=B TO END1
1090 IF E$(I)="END" THEN 1120
1100 PRINT #2:E$(I)
1110 NEXT I
1120 CLOSE #2
1130 GOTO 200
1140 END
```

Now if you key **E** to edit, answer record number 1, then key

This was record # 1
//

The file will now contain

This was record # 1
And this is # 2
This is the third record
END

If you had, however, keyed

This was record # 1
END

then the file would now contain

This was record # 1
END
This is the third record
END

In this latter entry, when the file is rewritten back to tape or disk, only two records will be written—the first one and the END record (since the **WRITE** command stops when an END record is encountered).

Action **W**, as mentioned earlier, will write the memory file to disk or tape in lines 290-440. Again, a disk file name is requested if the file is a disk file.

Action **L** lists from a beginning to an ending line number (lines 760-770) and delays by requiring a count from 1 to any number between 200 to 1000 before the next line is displayed. You can experiment with various delays to slow down or speed up the listing to the screen. Note that when an END record is found (line 890), the listing terminates as if the end of the file had been

found. If you know, however, that there are more records that you have "cut off" by keying an END record, you could replace that END with whatever you want by using the **E** command, and following it with **//** or **STOP**, as described earlier.

The **P** command does the same thing as the **LIST** command, but prints the records to a printer instead of displaying them on the screen.

The **C** command will check a file you have just written to cassette to verify that what is read back is the same as what is contained in memory.

Finally, the **S** command stops the program, obviously erasing whatever file has been stored in memory. Be sure that you have saved your file to disk or tape, using the **W** command, unless you don't care whether the data is saved or not.

EDIT2

```
100 REM * ************************
110 REM *                      *
120 REM *        EDIT2         *
130 REM *                      *
140 REM *     COPYRIGHT 1983   *
150 REM *   DONALD C. KREUTNER *
160 REM * ************************
170 OPTION BASE 1
180 DIM A$(500)
190 DIM A2$(500)
200 FOR I=1 TO 500
210 A$(I)="@"
220 NEXT I
230 LAST1=1
240 REM *****COMMANDS*****
250 INPUT "/":B$
260 IF B$="HELP" THEN 390
270 IF B$="C" THEN 200
280 IF B$="A" THEN 530
290 IF B$="D" THEN 780
300 IF B$="T" THEN 890
310 IF B$="L" THEN 1080
320 IF B$="K" THEN 1150
330 IF B$="LU" THEN 1300
340 IF B$="R" THEN 1370
350 IF B$="P" THEN 1550
360 IF B$="PU" THEN 1650
370 IF B$="E" THEN 1730
380 REM *****HELP SCREEN*****
390 PRINT "HELP PRINT THESE INSTRUCTIONS"
400 PRINT "A    ADD LINES"
410 PRINT "C    CLEAR WORKFILE"
420 PRINT "D    DELETE LINES"
430 PRINT "T    TEXT IN A FILE"
440 PRINT "L    LIST WORKFILE"
450 PRINT "K    KEEP A FILE"
460 PRINT "LU   LIST WORKFILE
                UNNUMBERED"
```

EDIT2 creates the same type of file as EDIT1. EDIT2, however, uses a different manner of manipulating the data, thus giving you a choice of "tools" to use in creating or altering data files.

Perhaps the principal difference is that EDIT2 adds records, using "steps" of record numbers. With this method, you can add records, starting at record number 120 and adding in steps of 5—that is, record number 125, then 130, 135, etc. Thus, gaps are left for adding records that need to be inserted later.

EDIT2 also can renumber records. For example, you can renumber the entire file in memory from record number 1 to 500 in any desired step (such as 1, 5, 9, etc., in steps of 4).

Two arrays are used to keep track of the data and renumber it (A$ and A2$ in lines 180-190). Notice that the **FOR-NEXT** loop in lines 200-230 puts an @ in every position of A$ so that any data added later can be easily identified from "empty" lines (ones that contain an @).

The different commands are listed on the menu screen (lines 240-500).

To add lines the **A** command branches to line number 530, where you specify the beginning line number and the step for incrementing line numbers. If the step is given as 1, then there will be no gaps between each record, unless you later renumber the file. One major difference between this program and EDIT1 is that if

```
470 PRINT "R      RENUMBER WORKFILE"
480 PRINT "P      PRINT WORKFILE"
490 PRINT "PU     PRINT WORKFILE
                  UNNUMBERED"
500 PRINT "E      END PROGRAM"
510 GOTO 250
520 REM ***ADD LINES***
530 INPUT "LINE#: ":A
540 INPUT "STEP#: ":STEP1
550 IF STEP1<>0 THEN 570
560 STEP1=5
570 IF A$(A)<>"@" THEN 740
580 IF A<10 THEN 620
590 IF A<100 THEN 640
600 IF A<1000 THEN 650
610 PRINT A;
620 PRINT "   ";A;
630 GOTO 680
640 PRINT "  ";A;
650 GOTO 680
660 PRINT " ";A;
670 GOTO 680
680 INPUT " ":C$
681 IF A<LAST1 THEN 700
690 LAST1=A
700 IF C$="END" THEN 760
710 A$(A)=C$
720 A=A+STEP1
730 GOTO 570
740 PRINT "THIS LINE ALREADY EXISTS"
750 PRINT A$(A)
760 GOTO 250
770 REM *****DELETE LINES*****
780 INPUT "BEG LINE#: ":B
790 INPUT "END LINE#: ":E
791 IF E<LAST1 THEN 800
792 LAST1=B
793 A$(B)="END"
800 IF E>500 THEN 790
810 FOR I=B TO E
```

you try to add a record on top of a record (or line) number that already has data in it, you are warned about the condition, and the program returns to the menu. Your options then are to renumber the data, leaving gaps, or to delete lines that you meant to get rid of.

You should note that when the program is **RUN**, a **/** prompt comes up on the screen. If you key **HELP** or any unacceptable command, the program will print a menu of commands available.

By keying **C** after the **/** prompt, you will clear the work file in memory, wiping out any file you may have "texted" (or read) in using the **T** command, or entered through the **A** command, as described earlier.

The **D** command will delete any desired lines from a beginning line number to an ending line number (lines 770-860). As mentioned earlier, you can delete lines within a specified range or from a certain line number to the end. (500 is the maximum line number.)

To "read in" a file from disk or tape, the **T** or **TEXT** command is used (lines 890-1060). You are asked for the step to be used in adding the new records to the work file.

Another major difference between EDIT1 and EDIT2 is that in EDIT2, records added from a file will not automatically erase previously existing data. The program keeps track of where the last record was and continues from that point. Of course, you can **CLEAR** the work file at any time by using the **C** command before texting in a file.

```
 820 IF A$(I)="@" THEN 860
 830 REM ***SAME AS ABOVE LINES***
 840 PRINT " ";A$(I)
 850 A$(I)="@"
 860 NEXT I
 870 GOTO 250
 880 REM *****TEXT IN FILE*****
 890 INPUT "CASSETTE/DISK (C/D): ":CD$
 900 IF CD$="D" THEN 940
 910 IF CD$<>"C" THEN 890
 920 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
 930 GOTO 960
 940 INPUT "INPUT FILE NAME: ":F$
 950 OPEN #1:F$
 960 INPUT "STEP#: ":STEP1
 970 IF STEP1<>0 THEN 990
 980 STEP1=5
 990 FOR I=LAST1 TO 500 STEP STEP1
1000 INPUT #1:A$(I)
1010 IF A$(I)="END" THEN 1040
1020 LAST1=I
1030 NEXT I
1040 CLOSE #1
1050 A$(I)="@"
1060 GOTO 250
1070 REM *****LIST WORKFILE*****
1080 FOR I=1 TO 500
1090 IF I>LAST1 THEN 1130
1100 IF A$(I)="@" THEN 1120
1110 PRINT I;A$(I)
1120 NEXT I
1130 GOTO 250
1140 REM *****KEEP AS FILE*****
1150 INPUT "CASSETTE/DISK (C/D): ":CD$
1160 IF CD$="D" THEN 1190
1170 OPEN #2:"CS1",OUTPUT,INTERNAL,
     FIXED 128
1180 GOTO 1210
1190 INPUT "OUTPUT FILE NAME: ":F$
```

(Continued in next column)

Two commands will list the records in the work file: the **L** and the **LU** commands. The **L** command will list the data immediately following the line number, whereas the **LU** command will list the data only. You can use the **L** command to determine what line numbers are available for your use.

The **P** and the **PU** commands are similar to **L** and **LU**, but print the data on a printer if you have one.

The **K** command "keeps" the work file in a disk or tape file, whereas the **E** command ends the program.

EDIT2

HELP	PRT INSTRUCTIONS
A	ADD LINES
C	CLEAR WORKFILE
D	DELETE LINES
T	TEXT IN FILE
L	LIST WORKFILE
K	KEEP A FILE
LU	LIST UNNUMBERED
R	RENUMBER WORKFILE
P	PRINT WORKFILE
PU	PRINT UNNUMBERED
E	END PROGRAM

(Continued from previous column)

```
1200 OPEN #2:F$
1210 FOR I=1 TO 500
1220 IF I>LAST1 THEN 1260
1230 IF A$(I)="@" THEN 1250
1240 PRINT #2:A$(I)
1250 NEXT I
1260 PRINT #2:"END"
```

```
1270 CLOSE #2
1280 GOTO 250
1290 REM *****LIST WORKFILE
     UNNUMBERED*****
1300 FOR I=1 TO 500
1310 IF I>LAST1 THEN 1350
1320 IF A$(I)="@" THEN 1340
1330 PRINT A$(I)
1340 NEXT I
1350 GOTO 250
1360 REM *****RENUMBER WORKFILE*****
1370 INPUT "STEP#: ":STEP1
1380 J=0
1390 IF STEP1<>0 THEN 1410
1400 STEP1=5
1410 FOR I=1 TO 500
1420 A2$(I)="@"
1430 NEXT I
1440 FOR I=1 TO 500
1450 IF I>LAST1 THEN 1500
1460 IF A$(I)="@" THEN 1490
1470 J=J+STEP1
1480 A2$(J)=A$(I)
1490 NEXT I
1500 FOR I=1 TO 500
1510 A$(I)=A2$(I)
1515 IF A$(I)="@" THEN 1520
1516 LAST1=I
1520 NEXT I
1530 GOTO 250
1540 REM *****PRINT WORKFILE*****
1550 OPEN #3:"RS232.BA=1200"
1560 FOR I=1 TO 500
1570 IF I>LAST1 THEN 1620
1580 IF A$(I)="@" THEN 1610
1590 REM ***VARIOUS LINES FOR LENGTH
     OF A***
1600 PRINT #3:I;TAB(5);A$(I)
1610 NEXT I
```

(Continued in next column)

(Continued from previous column)

```
1620 CLOSE #3
1630 GOTO 250
1640 REM *****PRINT WORKFILE
     UNNUMBERED*****
1650 OPEN #3:"RS232.BA=1200"
1660 FOR I=1 TO 500
1670 IF I>LAST1 THEN 1710
1680 IF A$(I)="@" THEN 1700
1690 PRINT #3:A$(I)
1700 NEXT I
1710 CLOSE #3
1720 GOTO 250
1730 END
```

EDITMASK

```
100 REM * ************************
110 REM *                        *
120 REM *        EDITMASK        *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 INPUT "ENTER # TO EDIT: ":N1
180 IF N1<>INT(N1)THEN 170
190 IF N1=9 THEN 580
200 NUMED=N1
210 MASK$="XX,XXX,XXX.XX-"
220 GOSUB 270
230 PRINT MASK$
240 GOTO 170
270 REM ***EDITING SUBROUTINE***
280 REM ***MASK$=MASK FOR EDITED RESULT
290 REM ***EXAMPLE: XXX,XXX.XX-
300 REM ***NUMED=NUMBER TO EDIT
    (INTEGER FORM)
310 NUMED2=NUMED
320 NUMED=ABS(NUMED)
330 LMASK=LEN(MASK$)
340 NUMED$=STR$(NUMED)
350 LNUMED=LEN(NUMED$)
360 CTN=LNUMED+1
370 FOR CTM=LMASK TO 1 STEP -1
380 CTN=CTN-1
390 IF CTN<1 THEN 550
400 IF SEG$(MASK$,CTM,1)<>"-" THEN 480
410 IF NUMED2>0 THEN 450
420 MASK$=SEG$(MASK$,1,CTM-1)&"-"
430 CTN=CTN+1
440 GOTO 470
450 MASK$=SEG$(MASK$,1,CTM-1)&" "
460 CTN=CTN+1
470 GOTO 560
```

EDITMASK is a fairly simple program that takes any number which you key (for example, 1200050) and edits it, putting in commas, a decimal point, and a minus sign, if needed. For the example just given, the edited result would be 12,000.50.

The importance of EDITMASK is to show how a formatted result can be obtained using regular TI BASIC, which does not have the **PRINT USING** option of the **PRINT** command, as does TI EXTENDED BASIC. With this subroutine any program can take a number, such as N1 in line 170, put it into NUMED, supply an editing "mask," such as MASK$ in line 210, and get the result in MASK$ by stating simply **GOSUB** 270, which means to perform the subroutine in line 270 until a **RETURN** statement is found.

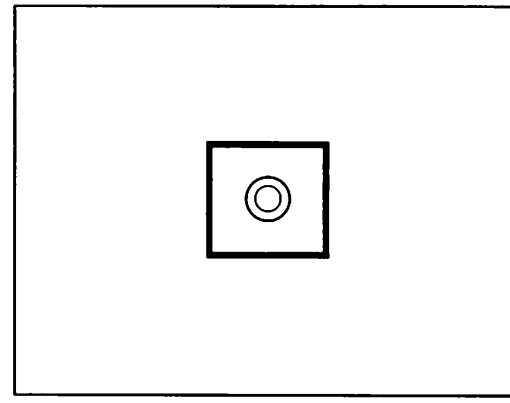
The subroutine looks from right to left through the editing mask and substitutes digits from the number NUMED. Zeros to the left are suppressed, along with unneeded commas, to obtain the result.

```
480 IF SEG$(MASK$,CTM,1)="," THEN 510
490 IF SEG$(MASK$,CTM,1)="." THEN 510
500 GOTO 530
510 CTN=CTN+1
520 GOTO 560
530 MASK$=SEG$(MASK$,1,CTM-1)&SEG$
    (NUMED$,CTN,1)&SEG$(MASK$,CTM+1,
    (LMASK-(CTM-1)-1))
540 GOTO 560
550 MASK$=SEG$(MASK$,1,CTM-1)&" "&SEG$
    (MASK$,CTM+1,(LMASK-(CTM-1)-1))
560 NEXT CTM
570 RETURN
580 END
```

FIGURE1

```
100 REM ************************
110 REM *                      *
120 REM *       FIGURE1        *
130 REM *                      *
140 REM *    COPYRIGHT 1983    *
150 REM *  DONALD C. KREUTNER  *
160 REM ************************
170 CALL CLEAR
180 INPUT "DELAY: ":DELAY
190 IF DELAY<0 THEN 170
200 IF DELAY=0 THEN 230
210 INPUT "PITCH (-1 TO -8, 110-44733): ":P
220 INPUT "VOLUME (0, LOUD TO 30, SOFT): ":V
230 REM FIGURE1
240 BC1$="4004"
250 BC2$="00000040000008"
260 BC3$="00000200000001"
270 BC4$="000100000002"
280 BC5$="0008000040"
290 BC6$="0000000000000440"
300 BC7$="0000000000004004"
310 BC8$="0010000004"
320 BC9$="008000000040"
330 BC10$="00004000000080"
340 BC11$="00000002000010"
350 BC12$="0220"
360 LC1$="4000080002000100"
370 LC2$="0001000200040040"
380 LC3$="0080004000100002"
390 LC4$="0200100040008000"
400 BLOCK$="FFFFFFFFFFFFFFFF"
410 CALL CHAR(91,BC1$)
420 CALL CHAR(92,BC2$)
430 CALL CHAR(93,BC3$)
440 CALL CHAR(94,BC4$)
450 CALL CHAR(95,BC5$)
460 CALL CHAR(96,BC6$)
```

FIGURE1 is a program that uses graphics to draw some very interesting patterns. Line 660 clears the screen. Then in the center of the screen a large circle is drawn clockwise with dots. Inside this circle another smaller circle is drawn. Finally, a square is drawn around the larger circle.



And yet that's not all. The figure of two circles and a square is repeated seven more times in symmetrical locations on the screen. Next, each group is "shot" in the middle with a solid dark square, and each square is then wiped out, with appropriate sound effects. The pattern is repeated over and over, creating an entertaining display for a computer show or just serving as a demonstration of what the TI-99/4A can do with graphics.

Lines 680-910 repeat the circles-and-squares patterns eight times with different coordinates designating rows (R) and columns (C). Special characters have been

```
470 CALL CHAR(97,BC7$)
480 CALL CHAR(98,BC8$)
490 CALL CHAR(99,BC9$)
500 CALL CHAR(100,BC10$)
510 CALL CHAR(101,BC11$)
520 CALL CHAR(102,BC12$)
530 CALL CHAR(103,LC1$)
540 CALL CHAR(104,LC2$)
550 CALL CHAR(105,LC3$)
560 CALL CHAR(106,LC4$)
570 CALL CHAR(107,"FFFFFFFF")
580 CALL CHAR(108,"FFFFFFFF0F0F0F0F")
590 CALL CHAR(109,"0F0F0F0F0F0F0F0F")
600 CALL CHAR(110,"0F0F0F0FFFFFFFFF")
610 CALL CHAR(111,"00000000FFFFFFFF")
620 CALL CHAR(112,"F0F0F0F0FFFFFFFF")
630 CALL CHAR(113,"F0F0F0F0F0F0F0F0")
640 CALL CHAR(114,"FFFFFFFFF0F0F0F0")
650 CALL CHAR(115,BLOCK$)
660 CALL CLEAR
670 PRINT TAB(11);"FIGURE1"
680 R=11
690 C=14
700 GOSUB 990
710 R=11
720 C=5
730 GOSUB 990
740 R=11
750 C=23
760 GOSUB 990
770 R=2
780 C=5
790 GOSUB 990
800 R=2
810 C=14
820 GOSUB 990
830 R=2
840 C=23
850 GOSUB 990
```

(Continued in next column)

defined by the **CHAR** statements in lines 410-650. The BC.$ fields represent "pieces" of the big circles, whereas the LC.$ fields are parts of the little circles. Lines 570-640 define pieces of the surrounding square. To determine what **CHAR** statements to use to draw such graphics, you can diagram a picture of what you want on graph paper. Then characters can be defined to obtain those diagrams.

Lines 920-950 shoot the squares into and out of the centers, and lines 960-970 delay the erasure before repeating the diagram.

At the beginning of the program, you are asked to supply a delay (to which the program must count after each portion is painted to the screen), a pitch, and a volume.

(Continued from previous column)

```
860 R=20
870 C=5
880 GOSUB 990
890 R=20
900 C=23
910 GOSUB 990
920 CHA=115
930 GOSUB 1540
940 CHA=32
950 GOSUB 1540
960 FOR I=1 TO 2000
970 NEXT I
980 GOTO 660
990 CALL VCHAR(R,C+2,91,1)
1000 GOSUB 1470
1010 CALL VCHAR(R,C+3,92,1)
```

```
1020 GOSUB 1470
1030 CALL VCHAR(R+1,C+3,93,1)
1040 GOSUB 1470
1050 CALL VCHAR(R+2,C+3,94,1)
1060 GOSUB 1470
1070 CALL VCHAR(R+3,C+3,95,1)
1080 GOSUB 1470
1090 CALL VCHAR(R+3,C+2,96,1)
1100 GOSUB 1470
1110 CALL VCHAR(R+3,C+1,97,1)
1120 GOSUB 1470
1130 CALL VCHAR(R+3,C,98,1)
1140 GOSUB 1470
1150 CALL VCHAR(R+2,C,99,1)
1160 GOSUB 1470
1170 CALL VCHAR(R+1,C,100,1)
1180 GOSUB 1470
1190 CALL VCHAR(R,C,101,1)
1200 GOSUB 1470
1210 CALL VCHAR(R,C+1,102,1)
1220 GOSUB 1470
1230 REM ***************************
1240 R=R+1
1250 C=C+1
1260 CALL VCHAR(R,C+1,103,1)
1270 GOSUB 1470
1280 CALL VCHAR(R+1,C+1,104,1)
1290 GOSUB 1470
1300 CALL VCHAR(R+1,C,105,1)
1310 GOSUB 1470
1320 CALL VCHAR(R,C,106,1)
1330 GOSUB 1470
1340 REM ***************************
1350 R=R-2
1360 C=C-2
1370 CALL HCHAR(R,C,107,6)
1380 CALL VCHAR(R,C+5,108,1)
1390 CALL VCHAR(R+1,C+5,109,5)
1400 CALL HCHAR(R+5,C+5,110,1)
```

(Continued in next column)

(Continued from previous column)

```
1410 CALL HCHAR(R+5,C,111,5)
1420 CALL HCHAR(R+5,C,112,1)
1430 CALL VCHAR(R,C,113,5)
1440 CALL HCHAR(R,C,114,1)
1450 GOSUB 1470
1460 RETURN
1470 IF DELAY<1 THEN 1510
1480 CALL SOUND(200,P,V)
1490 FOR I=1 TO DELAY
1500 NEXT I
1510 RETURN
1520 CALL SOUND(200,-7,20)
1530 RETURN
1540 CALL VCHAR(12,15,CHA,2)
1550 CALL VCHAR(12,16,CHA,2)
1560 GOSUB 1520
1570 CALL VCHAR(12,6,CHA,2)
1580 CALL VCHAR(12,7,CHA,2)
1590 GOSUB 1520
1600 CALL VCHAR(12,24,CHA,2)
1610 CALL VCHAR(12,25,CHA,2)
1620 GOSUB 1520
1630 CALL VCHAR(3,6,CHA,2)
1640 CALL VCHAR(3,7,CHA,2)
1650 GOSUB 1520
1660 CALL VCHAR(3,15,CHA,2)
1670 CALL VCHAR(3,16,CHA,2)
1680 GOSUB 1520
1690 CALL VCHAR(3,24,CHA,2)
1700 CALL VCHAR(3,25,CHA,2)
1710 GOSUB 1520
1720 CALL VCHAR(21,6,CHA,2)
1730 CALL VCHAR(21,7,CHA,2)
1740 GOSUB 1520
1750 CALL VCHAR(21,24,CHA,2)
1760 CALL VCHAR(21,25,CHA,2)
1770 GOSUB 1520
1780 RETURN
1790 END
```

GAMBLE

```
100 REM * ************************
110 REM *                        *
120 REM *        GAMBLE          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 OPTION BASE 1
180 DIM P(10)
190 DIM P$(10)
200 RANDOMIZE
210 DIE1=INT(6*RND+1)
220 DIE2=INT(6*RND+1)
230 CALL CLEAR
240 INPUT "# OF PLAYERS (1 TO 10): ":NP
250 IF NP<1 THEN 240
260 IF NP>10 THEN 240
270 INPUT "PLAY TO SCORE: ":PT
280 IF PT<1 THEN 270
290 REM ***************************
300 REM THIS GAME ALLOWS THE PLAYER
310 REM TO SHAKE THE DICE - THEN HE
320 REM HAS THE OPTION TO SHAKE
330 REM EITHER DIE AGAIN TO TRY TO
340 REM GET A DOUBLE - IF HE SHAKES
350 REM THE HIGHER DIE, HE HAS A
360 REM1 IN 2 CHANCE OF GETTING A
370 REM DOUBLE - BUT IF THE LOWER
380 REM DIE IS RESHAKEN, HE HAS A
390 REM1 IN 3 CHANCE
400 REM ***************************
410 REM SCORING:
420 REM NO DOUBLE - DICE TOTAL
430 REM DOUBLE - TWICE DICE TOTAL
440 REM FAILED DOUBLE - 0
450 REM ***************************
460 FOR I=1 TO NP
```

GAMBLE is a dice game that allows up to ten players to compete. The remarks in lines 290-440 briefly summarize the game's rules. Notice that after you shake the two dice, if you choose to reshake either die in an attempt to get a double, and you fail, you will get zero for your score on that attempt. The unique thing about this game is that it cannot be duplicated "as is" without a computer. To have one chance out of two to get a double with a real die, you would need to have three opportunities to try to match the other die. Similarly, to have one chance out of three, you would be allowed two reshakes, using real dice.

With GAMBLE, however, all you need is to tell the program you want to reshake die 1 or die 2 (line 710), or answer **N** to decline a rethrow. Then the program automatically determines whether you are rethrowing the higher or lower die, and computes your result, giving you the appropriate odds (lines 810 and 910). If you rethrow the lower die, line 810 selects a random number between 1 and 3. If the number 1 comes up, you have a double. The same logic applies to one chance out of two, except that you get a random result of 1 or 2.

In the program, lines 460-500 obtain the player names in the array P$ for as many players as are specified in line 240.

Each round of the game begins in line 550, and each player gets his turn in a long **FOR-NEXT** loop from line 560 to line 1450. When all players have had their turns, for one

```
470 PRINT "PLAYER #";I;"NAME:";
480 INPUT " ":P$(I)
490 P(I)=0
500 NEXT I
510 ROUND=0
520 REM FLAGD=0 MEANS NO DOUBLE
    ATTEMPT
530 REM FLAGD=1 MEANS LOW DIE DOUBLE
    ATTEMPT
540 REM FLAGD=2 MEANS HIGH DIE ATTEMPT
550 ROUND=ROUND+1
560 FOR I=1 TO NP
570 CALL CLEAR
580 PRINT "*****************************"
590 PRINT "ROUND # ";ROUND
600 PRINT "PLAYER #";I;
610 PRINT "(";P$(I);")"
620 PRINT
630 DIE2=INT(6*RND)+1
640 IF DIE1<>0 THEN 660
650 DIE1=1
660 IF DIE2<>0 THEN 680
670 DIE2=1
680 FLAGD=0
690 PRINT "DIE1=";DIE1;"  DIE2=";DIE2
700 PRINT
710 INPUT "RETHROW (N/1/2): ":N12$
720 IF N12$="N" THEN 1240
730 IF N12$="1" THEN 760
740 IF N12$="2" THEN 1010
750 GOTO 710
760 IF DIE1<=DIE2 THEN 790
770 FLAGD=2
780 GOTO 910
790 FLAGD=1
800 REM *************************
810 DIE1=INT(3*RND)+1
820 IF DIE1=1 THEN 880
830 IF DIE2=1 THEN 860
840 DIE1=DIE2-1
```

round, control returns from line 1460 to line 550, where the number of the round is incremented and another **FOR-NEXT** loop is started for a new round. This continues until one player reaches the goal for a specified number of points, selected in line 270.

Lines 590-700, using random number selections between 1 and 6, obtain the original results of throwing two dice.

Lines 810-990 allow an attempt to get a double using a rethrow of die 1. Line 760 analyzes whether that die is less than or equal to the other die. If you decide (stupidly) to reshake and you already have a double, you will have one chance out of two of getting the same double again. You should just answer **N** in line 710 if you get a double right away. (Remember that you get 10 points for any double whose point total doubled is less than 10.)

Lines 1000-1230 do the same thing for a reshake of die #2.

Lines 1240-1380 then figure your score. Lines 1271-1272 and lines 1321-1322 give you 10 points for a double that would otherwise score less than 10 points.

After each player's turn, the players' cumulative totals are displayed before the next player's turn (lines 1380-1450). If, however, any player's score exceeds the "play-to" score (PT), line 1390 transfers program control to line 1470, where the final scores are displayed.

```
 850 GOTO 870
 860 DIE1=DIE2+1
 870 GOTO 1240
 880 DIE1=DIE2
 890 GOTO 1240
 900 REM **************************
 910 DIE1=INT(2*RND)+1
 920 IF DIE1=1 THEN 980
 930 IF DIE2=1 THEN 960
 940 DIE1=DIE2-1
 950 GOTO 970
 960 DIE1=DIE2+1
 970 GOTO 1240
 980 DIE1=DIE2
 990 GOTO 1240
1000 REM **************************
1010 IF DIE2<=DIE1 THEN 1040
1020 FLAGD=2
1030 GOTO 1160
1040 FLAGD=1
1050 REM **************************
1060 DIE2=INT(3*RND)+1
1070 IF DIE2=1 THEN 1130
1080 IF DIE1=1 THEN 1110
1090 DIE2=DIE1-1
1100 GOTO 1120
1110 DIE2=DIE1+1
1120 GOTO 1240
1130 DIE2=DIE1
1140 GOTO 1240
1150 REM **************************
1160 DIE2=INT(2*RND)+1
1170 IF DIE2=1 THEN 1230
1180 IF DIE1=1 THEN 1210
1190 DIE2=DIE1-1
1200 GOTO 1220
1210 DIE2=DIE1+1
1220 GOTO 1240
1230 DIE2=DIE1
```

(Continued in next column)

(Continued from previous column)

```
1240 IF FLAGD=1 THEN 1310
1250 IF FLAGD=2 THEN 1310
1260 IF DIE1<>DIE2 THEN 1290
1270 TOT=2*(DIE1+DIE2)
1272 IF TOT>9 THEN 1280
1274 TOT=10
1280 GOTO 1350
1290 TOT=DIE1+DIE2
1300 GOTO 1350
1310 IF DIE1<>DIE2 THEN 1340
1320 TOT=2*(DIE1+DIE2)
1322 IF TOT>9 THEN 1330
1324 TOT=10
1330 GOTO 1350
1340 TOT=0
1350 P(I)=P(I)+TOT
1360 CALL CLEAR
1370 PRINT "DIE1=";DIE1;"   DIE2=";DIE2;
     TAB(23);P(I)
1380 PRINT "**************************"
1390 IF P(I)>=PT THEN 1470
1400 PRINT "****TOTALS****"
1410 FOR J=1 TO NP
1420 PRINT "PLAYER#";J;P$(J);TAB(23);P(J)
1430 NEXT J
1440 INPUT "ENTER TO CONTINUE: ":E$
1450 NEXT I
1460 GOTO 550
1470 CALL CLEAR
1480 PRINT "****FINAL SCORE****"
1490 FOR J=1 TO NP
1500 PRINT "PLAYER#";J;P$(J);TAB(23);P(J)
1510 NEXT J
1520 PRINT
1530 INPUT "PLAY AGAIN (Y/N): ":YN$
1540 IF YN$="Y" THEN 240
1550 END
```

GUESS3

```
100 REM * ************************
110 REM *                        *
120 REM *        GUESS3          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 RANDOMIZE
180 CALL CLEAR
190 PRINT "      **********"
200 PRINT "      **GUESS3**"
210 PRINT "      **********"
220 PRINT
230 PRINT
240 PRINT
250 PRINT
260 PRINT
270 PRINT
280 INPUT "PLAY OR QUIT (P/Q): ":PQ$
290 IF PQ$="Q" THEN 920
300 CALL CLEAR
310 X=INT(999*RND)+1
320 X$=STR$(X)
330 CTR=0
340 LENX=LEN(X$)
350 ON LENX GOTO 380,400,410
360 DISPLAY LENX
370 GOTO 280
380 X$="00"&X$
390 GOTO 410
400 X$="0"&X$
410 X1$=SEG$(X$,1,1)
420 X2$=SEG$(X$,2,1)
430 X3$=SEG$(X$,3,1)
440 CTR=CTR+1
450 PRINT "ATTEMPT#: ";CTR
460 INPUT "GUESS MY 3 DIGIT #: ":Y$
```

GUESS3 is a logic analysis game that allows you to try to guess a three-digit number based on the clues you obtain from previous guesses.

First, a random number is selected by the program from 1 to 999 (line 310). In lines 320-400, a one- or two-digit number has leading zeros added to the string representation of the number (X$ in line 320). Thus, the number 9 is 009, and 98 is 098. X1$, X2$, and X3$ contain the first, second, and third digits to be used in comparing to the guessed numbers. The number of guesses you make is incremented in line 440 prior to the three-digit guess of line 460.

If you failed to enter three digits, line 480 returns you to line 460 to reenter a new guess. (To guess the number 3, you must key **003**.) Then Y1$, Y2$, and Y3$ are given the values of the three guessed digits to be compared to the actual digits. Lines 570-640 check for the right numbers being in the right positions. If all three numbers are correct, you've guessed the random number, and control transfers from line 630 to line 740.

Otherwise, lines 650-700 check each digit you guessed against each digit of the number and tell you how many digits of your guess were correct (but not necessarily in the right position). From the accumulated information, you should try to guess the number in as few guesses as possible.

```
470 LENY=LEN(Y$)
480 IF LENY<>3 THEN 460
490 Y1$=SEG$(Y$,1,1)
500 Y2$=SEG$(Y$,2,1)
510 Y3$=SEG$(Y$,3,1)
520 CTRNUM=0
530 CTRPOS=0
540 X1F=0
550 X2F=0
560 X3F=0
570 IF X1$<>Y1$ THEN 590
580 CTRPOS=CTRPOS+1
590 IF X2$<>Y2$ THEN 610
600 CTRPOS=CTRPOS+1
610 IF X3$<>Y3$ THEN 650
620 CTRPOS=CTRPOS+1
630 IF CTRPOS=3 THEN 740
640 REM IF CTRPOS=3 YOU GOT IT!
650 Z$=Y1$
660 GOSUB 770
670 Z$=Y2$
680 GOSUB 770
690 Z$=Y3$
700 GOSUB 770
710 PRINT "YOU HAVE";CTRPOS;"(RIGHT
    POSITION)"
720 PRINT "AND ....";CTRNUM;"(RIGHT DIGITS)"
730 GOTO 440
740 PRINT "RIGHT!!!";CHR$(7);" IT'S ";X$
750 PRINT "**************************"
760 GOTO 280
770 IF X1F=1 THEN 820
780 IF Z$<>X1$ THEN 820
790 X1F=1
800 CTRNUM=CTRNUM+1
810 GOTO 910
820 IF X2F=1 THEN 870
830 IF Z$<>X2$ THEN 870
840 X2F=1
```

(Continued in next column)

```
ATTEMPT NUMBER 1
GUESS MY 3 DIGIT NUMBER: 123
YOU HAVE 0 RIGHT POSITION
AND... 1 RIGHT DIGITS
ATTEMPT NUMBER 2
GUESS MY 3 DIGIT NUMBER: 145
YOUR HAVE 1 RIGHT POSITION
AND... 2 RIGHT DIGITS
ATTEMPT NUMBER 3
GUESS MY 3 DIGIT NUMBER:
```

(Continued from previous column)

```
850 CTRNUM=CTRNUM+1
860 GOTO 910
870 IF X3F=1 THEN 910
880 IF Z$<>X3$ THEN 910
890 X3F=1
900 CTRNUM=CTRNUM+1
910 RETURN
920 END
```

GUESS4

```
100 REM * ************************
110 REM *                        *
120 REM *        GUESS4          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 RANDOMIZE
180 CALL CLEAR
190 PRINT "        **********"
200 PRINT "        **GUESS4**"
210 PRINT "        **********"
220 PRINT
230 PRINT
240 PRINT
250 PRINT
260 PRINT
270 PRINT
280 INPUT "PLAY OR QUIT (P/Q): ":PQ$
290 IF PQ$="Q" THEN 1050
300 CALL CLEAR
310 X=INT(9999*RND)+1
320 X$=STR$(X)
330 CTR=0
340 LENX=LEN(X$)
350 ON LENX GOTO 380,400,420,430
360 DISPLAY LENX
370 GOTO 280
380 X$="000"&X$
390 GOTO 430
400 X$="00"&X$
410 GOTO 430
420 X$="0"&X$
430 X1$=SEG$(X$,1,1)
440 X2$=SEG$(X$,2,1)
450 X3$=SEG$(X$,3,1)
```

(Continued in next column)

Although similar to GUESS3, GUESS4 requires you to guess a four-digit number instead of one with three digits.

In line 310 a random number between 1 and 9999 is selected. Leading zeros are added to numbers less than 1000 in lines 340-420.

The number you guess in line 490 is compared to the number to be guessed, just as in GUESS3. Obviously, you will need more guesses to discover a four-digit number than a three-digit number.

```
ATTEMPT NUMBER 1
GUESS MY 4 DIGIT NUMBER: 1234
YOU HAVE 0 RIGHT POSITION
AND... 2 RIGHT DIGITS
ATTEMPT NUMBER 2
GUESS MY 4 DIGIT NUMBER: 2156
YOUR HAVE 1 RIGHT POSITION
AND... 1 RIGHT DIGITS
ATTEMPT NUMBER 3
GUESS MY 4 DIGIT NUMBER:
```

(Continued from previous column)

```
460 X4$=SEG$(X$,4,1)
470 CTR=CTR+1
480 PRINT "ATTEMPT#: ";CTR
490 INPUT "GUESS MY 4 DIGIT #: ":Y$
500 LENY=LEN(Y$)
```

```
510 IF LENY<>4 THEN 490
520 Y1$=SEG$(Y$,1,1)
530 Y2$=SEG$(Y$,2,1)
540 Y3$=SEG$(Y$,3,1)
550 Y4$=SEG$(Y$,4,1)
560 CTRNUM=0
570 CTRPOS=0
580 X1F=0
590 X2F=0
600 X3F=0
610 X4F=0
620 IF X1$<>Y1$ THEN 640
630 CTRPOS=CTRPOS+1
640 IF X2$<>Y2$ THEN 660
650 CTRPOS=CTRPOS+1
660 IF X3$<>Y3$ THEN 680
670 CTRPOS=CTRPOS+1
680 IF X4$<>Y4$ THEN 720
690 CTRPOS=CTRPOS+1
700 IF CTRPOS=4 THEN 830
710 REMIF CTRPOS=4 YOU GOT IT!
720 Z$=Y1$
730 GOSUB 860
740 Z$=Y2$
750 GOSUB 860
760 Z$=Y3$
770 GOSUB 860
780 Z$=Y4$
790 GOSUB 860
800 PRINT "YOU HAVE";CTRPOS;"(RIGHT
    POSITION)"
810 PRINT "AND ....";CTRNUM;"(RIGHT DIGITS)"
820 GOTO 470
830 PRINT "RIGHT!!!";CHR$(7);" IT'S ";X$
840 PRINT "**************************"
850 GOTO 280
860 IF X1F=1 THEN 910
870 IF Z$<>X1$ THEN 910
880 X1F=1
```

(Continued in next column)

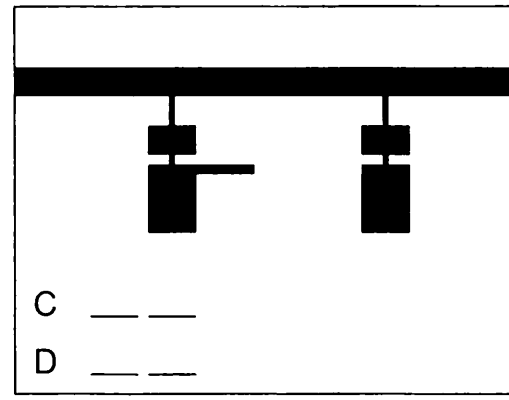
(Continued from previous column)

```
890 CTRNUM=CTRNUM+1
900 GOTO 1040
910 IF X2F=1 THEN 960
920 IF Z$<>X2$ THEN 960
930 X2F=1
940 CTRNUM=CTRNUM+1
950 GOTO 1040
960 IF X3F=1 THEN 1000
970 IF Z$<>X3$ THEN 1000
980 X3F=1
990 CTRNUM=CTRNUM+1
1000 IF X4F=1 THEN 1040
1010 IF Z$<>X4$ THEN 1040
1020 X4F=1
1030 CTRNUM=CTRNUM+1
1040 RETURN
1050 END
```

HANGMAN

```
100 REM * ************************* *
110 REM *                           *
120 REM *          HANGMAN          *
130 REM *                           *
140 REM *       COPYRIGHT 1983      *
150 REM *     DONALD C. KREUTNER    *
160 REM * ************************* *
170 RANDOMIZE
180 DIM X$(200)
190 DIM G1$(30)
200 DIM G2$(30)
210 REM HANGMAN
220 DATA LOUSY,WORDS,ANIMAL
230 DATA STRATEGY,PRODUCT,SYSTEM
240 DATA PROVIDE,FUTURE,DIFFICULT
250 DATA STORAGE,LIMITED,CLOSED
260 DATA IRON,STOVE,WASHER
270 DATA SWEATER,TABLE,COMPUTER
280 DATA USEFUL,MACHINE,SUBTRACT
290 DATA BASEBALL,ONIONS,WATERMELON
300 DATA LANTERN,PRIMARY,AWFUL
310 DATA FOREST,COUNTRY,BEAUTIFUL
320 DATA FOWL,ALPHABET,TEENAGER
330 DATA ELEPHANT,TELEVISION,REVISION
340 DATA TELEPHONE,TOASTER,FATHER
350 DATA END
360 REM YOU CAN USE THE WORDS
370 REM SUPPLIED OR REKEY NEW
380 REM ONES IN DIFFERENT DATA
390 REM STATEMENTS - OR YOU CAN
400 REM KEY EACH OTHER'S WORDS
410 BL$="FFFFFFFFFFFFFFFF"
420 RO$="1818181818181818"
430 TB$="1818FFFFFFFFFFFF"
440 AR$="0000FFFF00000000"
450 RL$="8040201008040201"
460 LL$="0102040810204080"
```

HANGMAN is a game for one or two players. With one player the words to be used are taken from the **DATA** statements in lines 220-350. With two players these preprogrammed words may be used, or you can make up your own words. When the preprogrammed words are used, the game randomly selects one or two of them, depending on the number of players. If you want, you can replace the words in the **DATA** statements with your own. Lines 620-650 load the array X$ with the words. Special block characters are defined to draw the two hangmen. Lines 410-600 define these characters. BL$ stands for a solid block used to draw the upper hanging support. RO$ is the rope, TB$ is the top body, AR$ is used for the arms, RL$ is the right leg, LL$ is the left leg, RF$ and LF$ are the feet, and RH$ and LH$ are the hands.



The two string arrays G1$ and G2$, defined in lines 190 and 200, are used to accumulate the past guesses of each person.

```
470 RF$="000000000000F0F0"
480 LF$="0000000000000F0F"
490 RH$="F0F0F0F000000000"
500 LH$="0F0F0F0F00000000"
510 CALL CHAR(97,BL$)
520 CALL CHAR(98,RO$)
530 CALL CHAR(99,TB$)
540 CALL CHAR(100,AR$)
550 CALL CHAR(101,RL$)
560 CALL CHAR(102,LL$)
570 CALL CHAR(103,RF$)
580 CALL CHAR(104,LF$)
590 CALL CHAR(105,RH$)
600 CALL CHAR(106,LH$)
610 CALL CLEAR
620 FOR I=1 TO 200
630 READ X$(I)
640 IF X$(I)="END" THEN 660
650 NEXT I
660 CTR=I-1
670 INPUT "1 OR 2 PLAYERS(99 TO QUIT): ":P
680 GUESS1=0
690 GUESS2=0
700 P1$=" "
710 P2$=" "
720 YN$=" "
730 IF P=99 THEN 2480
740 IF P<1 THEN 670
750 IF P>2 THEN 670
760 IF P<>2 THEN 840
770 INPUT "PLAYER1'S NAME: ":P1$
780 INPUT "PLAYER2'S NAME: ":P2$
790 INPUT "KEY EACH OTHER'S WORDS
    (Y/N): ":YN$
800 IF YN$<>"Y" THEN 840
810 INPUT "PLAYER1 KEY PLAYER2'S
    WORD: ":W2$
820 CALL CLEAR
830 INPUT "PLAYER2 KEY PLAYER1'S
    WORD: ":W$
```

With this feature you don't need to write down your guesses; the program remembers for you and tells you on each turn what guesses you have made so far.

If you choose the two-player game in line 670, the players' names are requested in lines 770 and 780. You can key each other's words or let the computer choose words at random (lines 790-890). Lines 900-1010 then create two strings of underscores, one for each letter in the two selected words.

Next, player 1 chooses a letter. If the letter "fits" for any letter(s) of the word selected for player 1 (W$), that letter replaces the underscore(s) in Y$, which is displayed after each turn to allow the players to see what letters they have correctly guessed so far. If a player guesses a letter in the hidden word, a switch (SW$) is set. If the switch is **Y**, ERR1 is not added to. Later, when the hangman is drawn, the number of errors made determines how "complete" a figure is—in other words, how completely a figure is hanging.

This process takes place for player 1 in lines 1030-1230, and for player 2 in lines 1240-1430. Both players have certain displays performed by the same routines, but each player is given different row and column coordinates. For instance, GOSUB 2040 in lines 1500 and 1780 draws the two hangmen, with the number of parts dependent on the number of errors made. Lines 1470-1490 and 1750-1770 define the base row and column, and the number of errors (R, C, and E). The guesses are displayed by lines 1510-1530 and lines 1790-1810. The players' names are printed by lines 1540-1560 and lines 1820-1840.

```
 840 IF YN$="Y" THEN 900
 850 A=INT(CTR*RND)+1
 860 B=INT(CTR*RND)+1
 870 W$=X$(A)
 880 W2$=X$(B)
 890 IF A=B THEN 860
 900 Y$="_"
 910 Y2$="_"
 920 ERR1=0
 930 ERR2=0
 940 LENW=LEN(W$)
 950 LENW2=LEN(W2$)
 960 FOR I=1 TO LENW-1
 970 Y$=Y$&"_"
 980 NEXT I
 990 FOR I=1 TO LENW2
1000 Y2$=Y2$&"_"
1010 NEXT I
1020 CALL CLEAR
1030 INPUT "GUESS A LETTER (PLAYER1): ":L$
1040 IF LEN(L$)=LENW THEN 2280
1050 GUESS1=GUESS1+1
1060 G1$(GUESS1)=L$
1070 IF LEN(L$)<>1 THEN 1030
1080 IF L$<"A" THEN 1030
1090 IF L$>"Z" THEN 1030
1100 SW$=" "
1110 FOR I=1 TO LENW
1120 IF L$<>SEG$(W$,I,1)THEN 1180
1130 IF I<>1 THEN 1160
1140 Y$=SEG$(W$,I,1)&SEG$(Y$,I+1,LENW-1)
1150 GOTO 1170
1160 Y$=SEG$(Y$,1,I-1)&SEG$(W$,I,1)&SEG$
     (Y$,I+1,LENW-I)
1170 SW$="Y"
1180 NEXT I
1190 CALL CLEAR
1200 IF SW$="Y" THEN 1220
1210 ERR1=ERR1+1
```

(Continued in next column)

HANGMAN continues until one of the players is "hung" or until one player has guessed all the letters in his word. Note that at any time you may spell out what you think your word is, but if you are wrong—you lose!

(Continued from previous column)

```
1220 GOSUB 1460
1230 IF P<>2 THEN 1030
1240 INPUT "GUESS A LETTER (PLAYER2): ":L$
1250 IF LEN(L$)=LENW2 THEN 2260
1260 GUESS2=GUESS2+1
1270 G2$(GUESS2)=L$
1280 IF LEN(L$)<>1 THEN 1240
1290 IF L$<"A" THEN 1240
1300 IF L$>"Z" THEN 1240
1310 SW$=" "
1320 FOR I=1 TO LENW2
1330 IF L$<>SEG$(W2$,I,1)THEN 1390
1340 IF I<>1 THEN 1370
1350 Y2$=SEG$(W2$,I,1)&SEG$(Y2$,I+1,LENW2-1)
1360 GOTO 1380
1370 Y2$=SEG$(Y2$,1,I-1)&SEG$(W2$,I,1)&SEG$
     (Y2$,I+1,LENW2-I)
1380 SW$="Y"
1390 NEXT I
1400 CALL CLEAR
1410 IF SW$="Y" THEN 1430
1420 ERR2=ERR2+1
1430 GOSUB 1460
1440 GOTO 1030
1450 CALL CLEAR
1460 CALL HCHAR(4,1,97,32)
1470 R=5
1480 C=7
```

```
1490 E=ERR1
1500 GOSUB 2040
1510 FOR I=1 TO GUESS1
1520 CALL VCHAR(13,I+2,ASC(G1$(I)),1)
1530 NEXT I
1540 FOR I=1 TO LEN(P1$)
1550 CALL VCHAR(12,I+2,ASC(SEG$(P1$,I,1)),1)
1560 NEXT I
1570 IF ERR1>9 THEN 2310
1580 COL=1
1590 FLAG1$="N"
1600 FOR I=1 TO LENW
1600 FOR I=1 TO LENW
1610 COL=COL+2
1620 CALL VCHAR(19,COL,ASC(SEG$(Y$,I,1)),1)
1630 IF SEG$(Y$,I,1)<>"_" THEN 1650
1640 FLAG1$="Y"
1650 NEXT I
1670 CALL VCHAR(20,4,ASC("L"),1)
1680 CALL VCHAR(20,5,ASC("A"),1)
1690 CALL VCHAR(20,6,ASC("Y"),1)
1700 CALL VCHAR(20,7,ASC("E"),1)
1710 CALL VCHAR(20,8,ASC("R"),1)
1720 CALL VCHAR(20,9,ASC("1"),1)
1730 IF FLAG1$="N" THEN 2420
1740 IF P=1 THEN 2020
1750 R=5
1760 C=25
1770 E=ERR2
1780 GOSUB 2040
1790 FOR I=1 TO GUESS2
1800 CALL VCHAR(16,I+2,ASC(G2$(I)),1)
1810 NEXT I
1820 FOR I=1 TO LEN(P2$)
1830 CALL VCHAR(15,I+2,ASC(SEG$(P2$,I,1)),1)
1840 NEXT I
1850 IF ERR2>9 THEN 2390
1860 COL=1
1870 FLAG1$="N"
```

(Continued in next column)

(Continued from previous column)

```
1880 FOR I=1 TO LENW2
1890 COL=COL+2
1900 CALL VCHAR(22,COL,ASC(SEG$(Y2$,I,1)),1)
1910 IF SEG$(Y2$,I,1)<>"_" THEN 1930
1920 FLAG1$="Y"
1930 NEXT I
1940 CALL VCHAR(23,3,ASC("P"),1)
1950 CALL VCHAR(23,4,ASC("L"),1)
1960 CALL VCHAR(23,5,ASC("A"),1)
1970 CALL VCHAR(23,6,ASC("Y"),1)
1980 CALL VCHAR(23,7,ASC("E"),1)
1990 CALL VCHAR(23,8,ASC("R"),1)
2000 CALL VCHAR(23,9,ASC("2"),1)
2010 IF FLAG1$="N" THEN 2450
2020 INPUT "RETURN TO CONTINUE: ":C$
2030 RETURN
2040 CALL HCHAR(R,C,98,1)
2050 IF E<1 THEN 2255
2060 CALL HCHAR(R+1,C,97,1)
2070 IF E<2 THEN 2255
2080 CALL HCHAR(R+2,C,99,1)
2090 CALL HCHAR(R+3,C,97,1)
2100 IF E<3 THEN 2255
2110 CALL HCHAR(R+2,C+1,100,1)
2120 IF E<4 THEN 2255
2130 CALL HCHAR(R+2,C-1,100,1)
2140 IF E<5 THEN 2255
2150 CALL HCHAR(R+4,C+1,101,1)
2160 IF E<6 THEN 2255
2170 CALL HCHAR(R+4,C-1,102,1)
2180 IF E<7 THEN 2255
2190 CALL HCHAR(R+2,C+2,105,1)
2200 IF E<8 THEN 2255
2210 CALL HCHAR(R+2,C-2,106,1)
2220 IF E<9 THEN 2255
2230 CALL HCHAR(R+4,C+2,103,1)
2240 IF E<10 THEN 2255
2250 CALL HCHAR(R+4,C-2,104,1)
2255 RETURN
```

```
2260 IF L$=W2$ THEN 2450
2270 GOTO 2390
2280 IF L$=W$ THEN 2420
2290 GOTO 2320
2310 REM
2320 PRINT "************************"
2330 PRINT "*P L A Y E R 1* LOSES*"
2340 PRINT "************************"
2350 PRINT W$
2360 PRINT W2$
2370 INPUT "RETURN TO CONTINUE: ":C$
2380 GOTO 670
2390 PRINT "************************"
2400 PRINT "*P L A Y E R 2* LOSES*"
2410 GOTO 2370
2420 PRINT "************************"
2430 PRINT "*P L A Y E R 1 WINS***"
2440 GOTO 2340
2450 PRINT "************************"
2460 PRINT "*P L A Y E R 2 WINS***"
2470 GOTO 2340
2480 END
```

LABEL1

```
100 REM * ************************
110 REM *                        *
120 REM *         LABEL1         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 REM EACH LABEL WILL BE PRINTED
180 REM FROM A SET OF 6 RECORDS
190 CALL CLEAR
200 PRINT "RC   READ CASSETTE FILE"
210 PRINT "RD   READ DISK FILE"
220 PRINT "END  END PROGRAM"
230 PRINT
240 INPUT "ACTION: ":A$
250 IF A$="RC" THEN 280
260 IF A$="RD" THEN 300
270 IF A$="END" THEN 440
280 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
290 GOTO 320
300 INPUT "DSK1.FILENAME: ":D$
310 OPEN #1:D$
320 OPEN #2:"RS232.BA=1200"
330 CTR=0
340 INPUT "START WITH RECORD#: ":NUM1
350 IF NUM1<1 THEN 340
360 INPUT #1:X$
370 CTR=CTR+1
380 IF X$="END" THEN 410
390 PRINT #2:X$
400 GOTO 360
410 CLOSE #1
420 CLOSE #2
430 GOTO 190
440 END
```

LABEL1 is one of the programs that uses input from either of the two edit programs, EDIT1 or EDIT2. Other programs in this chapter that also use EDIT1 or EDIT2 are SORT1 and RECIPE1. The two label programs and PRTRCMDS are the only programs in this book that require a printer. If you have a printer now or plan to get one soon, then these programs will be quite useful.

In this program, each label is printed on sticker or gummed labels. LABEL1 must read exactly six records for each label printed. When you key the data to be written to a file in EDIT1 or EDIT2, keep in mind that even if an address has less than six lines, you must still add blank records to "pad" each label to six records. The program allows you to start printing from a certain record in the file. Using the edit programs, you will know exactly which record number to start with if you do not want to print the entire file of labels.

LABELP

```
100 REM * ************************
110 REM *                        *
120 REM *        LABELP          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 REM ****************************
180 REM LABELP - PRINTS LABELS USING
190 REM UP TO 6 LINES AND REPEATS
200 REM ANY NUMBER OF LABELS FOR
210 REM THAT ADDRESS
220 REM ****************************
230 INPUT "NAME1 (XXXX TO QUIT): ":N1$
240 IF N1$="XXXX" THEN 420
250 INPUT "NAME2: ":N2$
260 INPUT "NAME3: ":N3$
270 INPUT "NAME4: ":N4$
280 INPUT "NAME5: ":N5$
290 INPUT "NAME6: ":N6$
300 INPUT "# OF LABELS: ":N1
310 OPEN #1:"RS232.BA=1200"
320 FOR I=1 TO N1
330 PRINT #1:N1$
340 PRINT #1:N2$
350 PRINT #1:N3$
360 PRINT #1:N4$
370 PRINT #1:N5$
380 PRINT #1:N6$
390 NEXT I
400 CLOSE #1
410 GOTO 230
420 END
```

LABELP is similar to LABEL1, but instead of reading a file of label input data, in LABELP you key the six lines of address information as the program requests it (lines 230-290). You are then asked how many labels you want with this keyed address. If you want 500 return address labels with your own address, you can easily accomplish this task with LABELP. If you want to send information to several addresses, you can make two labels of each, sending one and keeping the other to remind yourself of where you sent the information. Or if you do not need a record, you can simply request one label for each address keyed.

JOHN Q. DOE
1234 ANYROAD AVENUE
ANYTOWN, U.S.A. 99999

JOHN Q. DOE
1234 ANYROAD AVENUE
ANYTOWN, U.S.A. 99999

```
                LABEL P


NAME1 (XXXX TO QUIT):
NAME2:
NAME3:
NAME4:
NAME5:
NAME6:
NUMBER OF LABELS:
```

PRTRCMDS

```
100 REM * ************************
110 REM *                        *
120 REM *       PRTRCMDS         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 OPEN #1:"RS232.BA=1200"
180 CALL CLEAR
190 PRINT "CQ     CORRESPONDENCE
                  QUALITY"
200 PRINT "REG    REGULAR QUALITY"
210 PRINT "6LPI   6 LINES PER INCH"
220 PRINT "8LPI   8 LINES PER INCH"
230 PRINT "10CPI  10 CHARACTERS PER
                  INCH"
240 PRINT "12CPI  12 CHARACTERS PER
                  INCH"
250 PRINT "17CPI  17 CHARACTERS PER
                  INCH"
260 PRINT "DOUB   DOUB WIDTH
                  CHARACTERS"
270 PRINT "FF     FORMFEED (TOP
                  OF PAGE)"
280 PRINT "##PG   DESIGNATE PAGE LENGTH"
290 PRINT "PAGE   DEFAULT PAGE LENGTH"
295 PRINT "TYPE   TYPEWRITER MODE"
300 PRINT "END    END PROGRAM"
310 PRINT
320 INPUT "SELECTION: ":S$
330 IF S$="CQ" THEN 460
340 IF S$="REG" THEN 510
350 IF S$="6LPI" THEN 550
360 IF S$="8LPI" THEN 590
370 IF S$="10CPI" THEN 630
380 IF S$="12CPI" THEN 670
390 IF S$="17CPI" THEN 710
```

PRTRCMDS is a program that enables a printer to enter and leave certain modes of printing. When you use the TYPE command in PRTRCMDS, the printer also will perform like a typewriter. You can select many different print styles and then "type" the desired lines to your printer. PRTRCMDS has been designed for the control codes used on an Okidata Microline 92 printer, but some of the commands will work on many different models of printers.

Before you run PRTRCMDS, look in your printer's manual to see what character sequences cause your printer to do the things you want it to do. As stated in the Introduction, the printer statements used in this book assume a 1200-baud setting, but any other baud rate from 110 to 9600 is supported by the TI RS-232 interface. To adapt a program for use with a different printer, you may only need to change ".BA=1200" to ".BA=9600" or to some other setting.

In checking your printer's manual, you should find a control code reference section. Almost all printers use the FF (form feed) character to align the top of the page with the print head. In this program, FF is given in line 800. The ASCII code for FF is 12, so printing the function **CHR$(12)** to the printer will cause a form feed. Some commands that contain only one character being sent to the printer include FF; the 10CPI (10 characters per inch) command RS (ASCII 30); the 12CPI command FS

```
400 IF S$="DOUB" THEN 750
410 IF S$="FF" THEN 790
420 IF S$="PAGE" THEN 850
425 IF S$="TYPE" THEN 930
430 IF SEG$(S$,3,2)="PG" THEN 820
440 IF S$="END" THEN 970
450 GOTO 180
460 REM **CORR QUALITY**
470 GOSUB 880
480 PRINT #1:CHR$(27);"1"
490 GOSUB 900
500 GOTO 180
510 REM **REGULAR QUALITY**
520 GOSUB 880
530 GOSUB 900
540 GOTO 180
550 REM **6 LPI**
560 PRINT #1:CHR$(27);"6"
570 GOSUB 900
580 GOTO 180
590 REM **8 LPI**
600 PRINT #1:CHR$(27);"8"
610 GOSUB 900
620 GOTO 180
630 REM **10 CPI**
640 PRINT #1:CHR$(30)
650 GOSUB 900
660 GOTO 180
670 REM **12 CPI**
680 PRINT #1:CHR$(28)
690 GOSUB 900
700 GOTO 180
710 REM **17 CPI**
720 PRINT #1:CHR$(29)
730 GOSUB 900
740 GOTO 180
750 REM **DOUBLE WIDTH**
760 PRINT #1:CHR$(31)
770 GOSUB 900
780 GOTO 180
```

(ASCII 28); and the 17CPI command GS (ASCII 29). Printing of regular quality is caused by the CAN character (ASCII 24), which returns the printer to its default values.

Double-width characters are started by sending the US (ASCII 31) code to the printer.

Many printer commands are called "escape sequences." In other words, the printer escapes from its printing responsibility to perform a setup function.

Escape functions used in PRTRCMDS include the following:

1. Correspondence-quality printing is obtained by printing the escape sequence ESC-1 (ASCII 27, or escape; followed by ASCII 49, the ASCII number for the character "1").
2. 6LPI (6 lines per inch) is designated by ESC-6 (ASCII 27, or escape, and the character "6").
3. 8LPI is caused by the ESC-8 sequence.
4. A different page length is caused by sending the sequence ESC-F, followed by a two-digit number that represents the page length.

After many of these commands have been given to the printer, two lines of X's and a test pattern can be printed in lines 900-910 to demonstrate what mode the printer is now in. You can, for example, run this program to cause better quality printing

```
790 REM **TOP OF FORM**
800 PRINT #1:CHR$(12)
810 GOTO 180
820 REM **NEW PAGE LENGTH**
830 PRINT #1:CHR$(27);"F";SEG$(S$,1,2)
840 GOTO 790
850 REM **DEFAULT PAGE LENGTH**
860 PRINT #1:CHR$(27);"F";"66"
870 GOTO 790
880 PRINT #1:CHR$(24)
890 RETURN
900 INPUT "PRINT TEST PATTERN (Y/N): ":
    TPYN$
902 IF TPYN$<>"Y" THEN 920
904 PRINT #1:"XXXXXXXXXXXXXXXXXXX"
906 PRINT #1:"XXXXXXXXXXXXXXXXXXXX"
908 PRINT #1:"ABCDEFGHIJKLMNOPQRSTUVW
    XYZ0123456789"
910 PRINT #1:"!@#$%^&*()+=-/:;<,>.9[]?'|{}\`"
920 RETURN
930 PRINT "TYPE LINE (END TO STOP):"
935 INPUT "-":I$
940 IF I$="END" THEN 180
950 PRINT #1:I$
960 GOTO 930
970 CLOSE #1
980 END
```

before running another program that uses the printer. Or the program that does the printing can send the printer the appropriate codes immediately after opening the printer file (RS232.BA=1200 in all the programs in this book).

CQ	CORRESPONDENCE QUALITY
REG	REGULAR QUALITY
6LPI	6 LINES PER INCH
8LPI	8 LINES PER INCH
10CPI	10 CHARS PER INCH
12CPI	12 CHARS PER INCH
17CPI	17 CHARS PER INCH
DOUB	DOUBLE WIDTH CHARS
FF	FORMFEED (TOP OF PAGE)
##PG	DESIGNATE PAGE LENGTH
PAGE	DEFAULT PAGE LENGTH
END	END PROGRAM

```
XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
THE ABOVE IS REGULAR QUALITY
```

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
AND THIS IS CORRESPONDENCE
QUALITY; REMEMBER THE
TEST PATTERN IS OPTIONAL
WHEN CHANGING PRTR CMDS
```

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
THIS IS 10CPI (CHARS PER INCH)
```

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
AND THIS IS 12CPI
```

```
AND DOUBLE WIDTH
ABCDEFGHIJKLMNOPQR
STUVWXYZ0123456789
```

```
      XXXXXXXXXXXXXXXXXXX
      XXXXXXXXXXXXXXXXXXX
      ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
      !@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
      17CPI WORKS ONLY IN REGULAR MODE
```

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
ABOVE IS 6LPI (LINES PER INCH)
```

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
!@#$%^&*()+=-/:;<,>.~[]_?'|{}\`
AND 8LPI...
```

RECIPE1

```
100 REM * ************************
110 REM *                        *
120 REM *        RECIPE1         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 DIM N$(300)
180 CTR=0
190 DATA 01. CHOCOLATE CAKE
200 DATA CHOPPED WALNUTS/1/C
210 DATA FLOUR/2/C
220 DATA CHOCOLATE/.5/C
230 DATA SUGAR/1/C
240 DATA BLEND FLOUR WATER & SUGAR
250 DATA WATER/1/C
260 DATA END RECIPE
270 DATA 02. SUGAR COOKIES
280 DATA FLOUR/2/C
290 DATA SUGAR/1/C
300 DATA WATER/1/C
310 DATA END RECIPE
320 DATA END
330 CALL CLEAR
340 INPUT "DATA FROM PROGRAM OR FILE
    (P/F): ":PF$
350 IF PF$="P" THEN 380
360 IF PF$="F" THEN 430
370 GOTO 180
380 READ NAME$
390 CTR=CTR+1
400 N$(CTR)=NAME$
410 IF NAME$="END" THEN 590
420 GOTO 350
430 INPUT "CASSETTE OR DISK (C/D): ":CD$
440 IF CD$="C" THEN 470
450 IF CD$="D" THEN 490
```

RECIPE1 is a simple, yet versatile program that can be used to save ingredients and their quantities, as well as detailed preparation instructions for recipes or other types of "formula" applications, such as paint blending or chemical mixing.

With RECIPE1 you can also specify any multiple of the original recipe and thus obtain the amounts required for its ingredients.

Two basic functions are available: to list the menu of recipes that can be displayed in detail, and to print the specific ingredients and instructions for one recipe at a time.

The information can come from **DATA** statements contained within the program (lines 190-320). You can have different programs with different menus using different **DATA** statements.

Perhaps a more practical way to save your recipe information is to have only one program version (with certain frequently used recipes "built in" to **DATA** statements), and to read any desired file containing recipes. With this method, you can specify in line 340 whether you want to use the data in the program or in a file. Then, if you select a file, line 430 asks you whether the file is on cassette tape or disk. If the program data is chosen, lines 380-410 read the information from the **DATA** statements into the array N$, which can hold up to 300 lines of data. If a file is chosen, lines 510-570 read the data from the selected file into the same N$ array.

```
460 GOTO 430
470 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
480 GOTO 510
490 INPUT "DSK1.FILENAME: ":D$
500 OPEN #1:D$,INPUT
510 FOR I=1 TO 300
520 INPUT #1:NAME$
530 CTR=I
540 N$(I)=NAME$
550 IF NAME$="END" THEN 570
560 NEXT I
570 CLOSE #1
580 GOTO 590
590 PRINT "M     MENU OF RECIPES"
600 PRINT "##.   PRINT RECIPE ##"
610 PRINT "END  END PROGRAM"
620 PRINT
630 PRINT
640 PRINT
650 INPUT "SELECTION: ":S$
660 IF S$="M" THEN 700
670 IF SEG$(S$,3,1)="." THEN 760
680 IF S$="END" THEN 1150
690 GOTO 650
700 CALL CLEAR
710 FOR I=1 TO CTR
720 IF SEG$(N$(I),3,1)<>"." THEN 740
730 PRINT N$(I)
740 NEXT I
750 GOTO 590
760 FOR I=1 TO CTR
770 IF S$<>SEG$(N$(I),1,3)THEN 800
780 REC$=N$(I)
790 GOTO 830
800 NEXT I
810 PRINT "RECIPE #: ";S$;" NOT FOUND"
820 GOTO 590
830 INPUT "MULTIPLE OF RECIPE: ":MU
840 CALL CLEAR
850 PRINT REC$;" ";MU;"(MULT)"
```

Once the data has been stored in the array, its manipulation by the program is a simple matter. To list a menu of the available recipes loaded, just key **M** to the menu screen, and lines 700-740 read through the array, looking for lines that have a period in the third position. These lines are then listed (the "title" lines for the recipe records that follow them).

If you request a printout for a specific recipe, the program looks through the file for the title record, starting with the same ## and period (.) that you requested. The multiple you specify in line 830 is then multiplied by the quantities of each ingredient to get the totals needed.

Remember that in the programs EDIT1 and EDIT2, RECIPE1 was mentioned as one of the programs that could use data keyed by either of these two programs. Just make certain when you create a file (or **DATA** statements) that you begin each recipe with a **##.** and follow the period with a space and then the title of the recipe.

Then records for ingredients or preparation instructions can follow in any order since the RECIPE1 program "knows" which record is which. If there is no slash (/) in the line, the record is an instruction line, whereas ingredient lines have slashes separating the ingredient from the quantity and the quantity from the "unit of measure" field (for example, C for cup, T for tablespoon, or any other measurements you may want to use).

Each recipe must end with an "END RECIPE" line to tell the program when to stop listing the recipe. The "END" **DATA** statement in line 320 is necessary to tell the

```
 860 FOR J=I+1 TO CTR
 870 IF N$(J)="END RECIPE" THEN 1130
 880 LENS=LEN(N$(J))
 890 LOCSL1=0
 900 LOCSL2=0
 910 FOR L=1 TO LENS
 920 IF SEG$(N$(J),L,1)<>"/" THEN 970
 930 IF LOCSL1<>0 THEN 960
 940 LOCSL1=L
 950 GOTO 970
 960 LOCSL2=L
 970 NEXT L
 980 IF LOCSL1=0 THEN 1110
 990 INGRED$=SEG$(N$(J),1,LOCSL1-1)
1000 IF LOCSL1<>0 THEN 1050
1010 IF LOCSL2<>0 THEN 1050
1020 QTY=0
1030 MEAS$=" "
1040 GOTO 1090
1050 QTY$=SEG$(N$(J),LOCSL1+1,LOCSL2-
     LOCSL1-1)
1060 QTY=VAL(QTY$)
1070 MEAS$=SEG$(N$(J),LOCSL2+1,LENS-
     LOCSL2)
1080 QTY=QTY*MU
1090 PRINT INGRED$;TAB(20);QTY;TAB(25);MEAS$
1100 GOTO 1120
1110 PRINT N$(J)
1120 NEXT J
1130 INPUT "CONTINUE (Y/N): ":YN$
1140 IF YN$="Y" THEN 590
1150 END
```

program when to stop reading data. If reading from a file, the EDIT1/EDIT2 programs automatically put an "END" record at the end of the file.

RECIPE1

01. ROAST BEEF AU GRATIN
02. BARBEQUED CHICKEN
03. SWEDISH MEATBALLS
04. PIZZA
05. BEEF STROGANOFF

02.	Sugar Cookies	2	(Mult)
Flour		4	C
Sugar		2	C
Water		2	C

Here is a cookie recipe printout, but don't try it—I made it up!

SCREEN1

```
100 REM * ************************
110 REM *                        *
120 REM *        SCREEN1         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 CALL CLEAR
180 REM ***SCREEN1***
190 INPUT "REPEAT 1 LINE OR 5 (1/5): ":R$
200 IF R$="1" THEN 230
210 IF R$="5" THEN 440
220 GOTO 170
230 INPUT "MSG: ":A$
240 X=LEN(A$)
250 IF X>28 THEN 230
260 INPUT "#OF LINES: ":REPS
270 INPUT "#SCREENS: ":S
280 FOR K=1 TO S
290 CALL CLEAR
300 FOR J=1 TO REPS
310 FOR I=1 TO X
320 CALL VCHAR(J+1,I+1,ASC(SEG$(A$,I,1)))
330 NEXT I
340 NEXT J
350 FOR J=REPS TO 1 STEP -1
360 FOR I=X TO 1 STEP -1
370 CALL VCHAR(J+1,I+1,32)
380 NEXT I
390 NEXT J
400 NEXT K
410 INPUT "MORE (Y/N): ":YN$
420 IF YN$="Y" THEN 170
430 GOTO 930
440 INPUT "MSGA: ":A$
450 XA=LEN(A$)
460 IF XA>28 THEN 440
```

SCREEN1 is an interesting message program. If you have to leave at 4:45 p.m., and a family member is coming home at 5:00 p.m., just start up this program. Your relative can read a message (of up to five lines) that fills and clears the screen repeatedly.

Two variations are available: to repeat one line any number of times or to repeat five lines any number of times. If you select the one-liner, the screen will fill up with that line, one character at a time, and then erase, one character at a time, from right to left and bottom to top—unusual, to say the least!

The one-line variation is in lines 230-400, while the five-line message is in lines 440-910. The **VCHAR** call is used to move one character at a time from the string(s) you keyed as messages to be displayed.

```
SCREEN1 WILL DISPLAY
A MSG ON THE SCREEN OF UP TO
5 LINES 1 CHARACTER AT A TIME
AND THEN ERASE THE SCREEN
AND REPEAT THE MSG DISPLAY
```

```
470 INPUT "MSGB: ":B$
480 XB=LEN(B$)
490 IF XB>28 THEN 470
500 INPUT "MSGC: ":C$
510 XC=LEN(C$)
520 IF XC>28 THEN 500
530 INPUT "MSGD: ":D$
540 XD=LEN(D$)
550 IF XD>28 THEN 530
560 INPUT "MSGE: ":E$
570 XE=LEN(E$)
580 IF XE>28 THEN 560
590 INPUT "CONTINUE: ":YN2$
600 IF YN2$<>"Y" THEN 170
610 INPUT "DELAY: (0-1000) ":DELAY
620 INPUT "# OF REPITITIONS? ":R2
630 FOR K=1 TO R2
640 CALL CLEAR
650 IF XA<1 THEN 690
660 FOR I=1 TO XA
670 CALL VCHAR(2,I+1,ASC(SEG$(A$,I,1)))
680 NEXT I
690 IF XB<1 THEN 730
700 FOR I=1 TO XB
710 CALL VCHAR(3,I+1,ASC(SEG$(B$,I,1)))
720 NEXT I
730 IF XC<1 THEN 770
740 FOR I=1 TO XC
750 CALL VCHAR(4,I+1,ASC(SEG$(C$,I,1)))
760 NEXT I
770 IF XD<1 THEN 810
780 FOR I=1 TO XD
790 CALL VCHAR(5,I+1,ASC(SEG$(D$,I,1)))
800 NEXT I
810 IF XE<1 THEN 850
820 FOR I=1 TO XE
830 CALL VCHAR(6,I+1,ASC(SEG$(E$,I,1)))
840 NEXT I
850 FOR I=7 TO 10
```

(Continued in next column)

(Continued from previous column)

```
860 CALL HCHAR(I,2,42,28)
870 NEXT I
880 IF DELAY<1 THEN 910
890 FOR I=1 TO DELAY
900 NEXT I
910 NEXT K
920 GOTO 410
930 END
```

SORT1

```
100 REM * ************************
110 REM *                        *
120 REM *          SORT1         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 OPTION BASE 1
180 DIM BEG(10)
190 DIM LENG(10)
200 DIM X$(500)
210 CALL CLEAR
220 PRINT "R    READ FILE"
230 PRINT "S    SORT FILE"
240 PRINT "W    WRITE FILE"
250 PRINT "L    LIST FILE"
260 PRINT "E    END PROGRAM"
270 PRINT
280 INPUT "ACTION: ":A$
290 IF A$="R" THEN 340
300 IF A$="S" THEN 500
310 IF A$="W" THEN 1090
320 IF A$="L" THEN 890
330 IF A$="E" THEN 1220
340 CALL CLEAR
350 INPUT "READ FROM CASSETTE OR DISK
    (C/D): ":CD$
360 IF CD$="C" THEN 420
370 IF CD$="D" THEN 390
380 GOTO 340
390 INPUT "DSK1.FILENAME: ":D$
400 OPEN #1:D$
410 GOTO 430
420 OPEN #1:"CS1",INTERNAL,INPUT ,FIXED 128
430 FOR I=1 TO 498
440 INPUT #1:X$(I)
450 IF X$(I)="END" THEN 470
```

SORT1 is a versatile tool that can sort any file entered or modified by EDIT1 or EDIT2. In using SORT1 you can specify a sort key composed of up to ten fields in any positions of the record.

For example, let's consider a record composed of an account number of four digits; a name field of 20 positions, with last name first; a telephone number field of ten positions, with hyphens included; and an outstanding balance field, with leading zeros and a total of eight digits. Let's say that each field is separated from the previous field by a space. The record layout can be described as follows:

Pos. 01-04 Acct Number
Pos. 06-25 Last-First Name
Pos. 27-36 Telephone #
Pos. 38-45 Balance Due

Although this record provides a very simple example, you can use many more fields in the records keyed by EDIT1 or EDIT2. But even with records as short as this one, SORT1 enables you to sort, resort, and print (on the screen or a printer) data from this kind of record in many ways:

By account number only
By balance due by account number
By last name by account number
By telephone number

Note one final caution about entering the data: be sure to include the appropriate spaces in every field of every record. In the example above, all account numbers are

```
460 NEXT I
470 CTR=I-1
480 CLOSE #1
490 GOTO 210
500 J=0
510 INPUT "ASCENDING OR DESCENDING
    (A/D): ":AD$
520 J=J+1
530 PRINT "SORT KEY #";J
540 INPUT "BEG LOC OF FIELD(999 TO
    STOP): ":B
550 IF B=999 THEN 600
560 INPUT "LENGTH OF FIELD: ":L
570 BEG(J)=B
580 LENG(J)=L
590 GOTO 520
600 CTRS=J-1
610 PASS1=0
620 SW$="N"
630 PASS1=PASS1+1
640 CALL CLEAR
650 PRINT "SORT PASS";PASS1
660 FOR I=1 TO CTR-1
670 S1$=" "
680 S2$=" "
690 FOR J=1 TO CTRS
700 S1$=S1$&SEG$(X$(I),BEG(J),LENG(J))
710 S2$=S2$&SEG$(X$(I+1),BEG(J),LENG(J))
720 NEXT J
730 IF AD$="D" THEN 800
740 IF S1$<=S2$ THEN 850
750 X1$=X$(I)
760 X$(I)=X$(I+1)
770 X$(I+1)=X1$
780 SW$="Y"
790 GOTO 850
800 IF S1$>=S2$ THEN 850
810 X1$=X$(I)
820 X$(I)=X$(I+1)
830 X$(I+1)=X1$
```

four characters in length, followed by a space; all name fields are 20 characters long (even if the field requires ten or more trailing spaces), followed by a space; and so on. In short, each record keyed by the edit program should have fields located directly beneath the corresponding fields of the last record.

When you run the SORT1 program, the following options will appear on the menu screen: **READ** a file, **SORT** a file, **WRITE** a sorted file, **LIST** a sorted file to the screen or to a printer, or **END** the program. These functions are accomplished in lines 280-330 by keying **R**, **S**, **W**, **L**, or **E**.

The file can be read in from cassette or disk in lines 350-480.

A sort is accomplished by lines 500-870. Line 510 allows you to specify whether you want the sort to be in ascending or descending sequence. If you sort by amount due, you may want the largest amounts to appear first. Then lines 500-590 allow you to describe the beginning location of each sort field and the length of each. For example, to sort by last name by account number, you specify position 6 for the starting location of sort field 1, with a length of 20; and position 1 for the location of sort field 2, with a length of 4. Then you key **999** in line 540 for the third sort field (to indicate that there are no more fields).

The actual sorting takes place in lines 620-860. Lines 690-710 "construct" the sort field from the subsort fields you defined. Then lines 730-840 switch the data elements if they are out of sequence (for ascending or descending sequences).

```
 840 SW$="Y"
 850 NEXT I
 860 IF SW$="Y" THEN 620
 870 PRINT "SORT COMPLETE"
 880 GOTO 210
 890 CALL CLEAR
 900 INPUT "PRINTER (Y/N): ":PYN$
 905 INPUT "TITLE: ":T$
 910 IF PYN$<>"Y" THEN 930
 920 OPEN #2:"RS232.BA=1200"
 930 INPUT "PAUSE (Y/N): ":PA$
 940 CTRP=0
 944 PRINT T$
 945 PRINT
 946 IF PYN$<>"Y" THEN 950
 947 PRINT #2:T$
 948 PRINT #2
 950 FOR I=1 TO CTR
 960 PRINT X$(I)
 970 CTRP=CTRP+1
 980 IF PA$<>"Y" THEN 1020
 990 IF CTRP<20 THEN 1020
1000 INPUT "ENTER TO CONTINUE: ":E$
1010 CTRP=0
1020 IF PYN$<>"Y" THEN 1040
1030 PRINT #2:X$(I)
1040 NEXT I
1050 IF PYN$<>"Y" THEN 1070
1060 CLOSE #2
1070 INPUT "ENTER TO CONTINUE: ":E$
1080 GOTO 210
1090 INPUT "CASSETTE OR DISK (C/D): ":CD$
1100 IF CD$="D" THEN 1140
1110 IF CD$<>"C" THEN 1090
1120 OPEN #1:"CS1",OUTPUT,INTERNAL,
     FIXED 128
1130 GOTO 1160
1140 INPUT "DSK1.FILENAME: ":D$
1150 OPEN #1:D$
```

(Continued in next column)

Lines 900-1060 display the sorted data on the screen and/or to a printer.

Finally, in lines 1090-1200 the sorted file can be saved to a new file or to the same file as was input, if desired.

(Continued from previous column)

```
1160 FOR I=1 TO CTR
1170 PRINT #1:X$(I)
1180 NEXT I
1190 PRINT #1:"END"
1200 CLOSE #1
1210 GOTO 210
1220 END
```

UNSORTED DATA AS READ FROM FILE

ACCT	NAME(LAST FIRST)	TELEPHONE	BAL DUE
0001	Jones Robert	588-9912	00080000
0003	Edwards Barnaby	466-2314	00004556
0002	Kramer James	233-1254	00000000
0210	White Mike	555-1234	00009803
0032	Hanferd Hank	324-9983	00000300
0021	Smith Sam	432-1256	00021025

DATA SORTED BY ACCOUNT NUMBER (ASCENDING)

0001	Jones Robert	588-9912	00080000
0002	Kramer James	233-1254	00000000
0003	Edwards Barnaby	466-2314	00004556
0021	Smith Sam	432-1256	00021025
0032	Hanferd Hank	324-9983	00000300
0210	White Mike	555-1234	00009803

DATA SORTED BY BAL DUE BY ACCT# (DESCENDING)

0001	Jones Robert	588-9912	00080000
0021	Smith Sam	432-1256	00021025
0210	White Mike	555-1234	00009803
0003	Edwards Barnaby	466-2314	00004556
0032	Hanferd Hank	324-9983	00000300
0002	Kramer James	233-1254	00000000

DATA SORTED BY ACCOUNT NUMBER (DESCENDING)

0210	White Mike	555-1234	00009803
0032	Hanferd Hank	324-9983	00000300
0021	Smith Sam	432-1256	00021025
0003	Edwards Barnaby	466-2314	00004556
0002	Kramer James	233-1254	00000000
0001	Jones Robert	588-9912	00080000

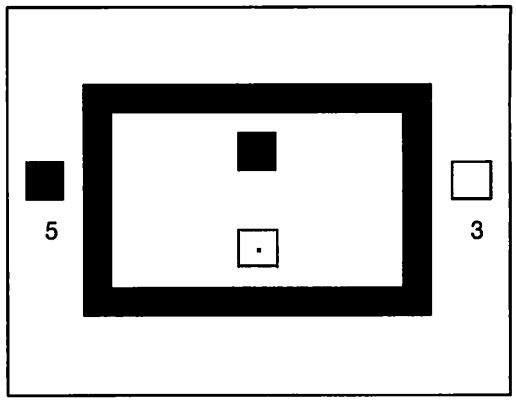
DATA SORTED BY LAST NAME BY ACCT#

0003	Edwards Barnaby	466-2314	00004556
0032	Hanferd Hank	324-9983	00000300
0001	Jones Robert	588-9912	00080000
0002	Kramer James	233-1254	00000000
0021	Smith Sam	432-1256	00021025
0210	White Mike	555-1234	00009803

```
DATA SORTED BY TELEPHONE#

0002        Kramer James        233-1254        00000000
0032        Hanferd Hank        324-9983        00000300
0021        Smith Sam           432-1256        00021025
0003        Edwards Barnaby     466-2314        00004556
0210        White Mike          555-1234        00009803
0001        Jones Robert        588-9912        00080000
```

TAG

```
100 REM * ************************
110 REM *                        *
120 REM *          TAG           *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 CALL CLEAR
180 REM ***TAG***
190 INPUT "MAX MOVES: ":MAX
200 ADD1$=" "
210 DOT$="W"
220 A$="FFFFFFFFFFFFFFFF"
230 B$="FF818181818181FF"
240 AD$="FFFFFFE7E7FFFFFF"
250 BD$="FF818199998181FF"
260 CALL CHAR(96,A$)
270 CALL CHAR(97,B$)
280 CALL CHAR(98,AD$)
290 CALL CHAR(99,BD$)
300 SCOREA=0
310 SCOREB=0
320 GOSUB 940
330 COUNT=0
340 RA=3
350 CA=16
360 RA2=RA
370 CA2=CA
380 RB=21
390 CB=16
400 RB2=RB
410 CB2=CB
420 GOSUB 1000
430 GOSUB 1200
440 GOSUB 1400
450 IF ADD1$="A" THEN 1530
460 IF COUNT>=MAX THEN 1530
```



TAG, as the name implies, is a game in which two players try to catch one another, each taking turns at being "it." At the beginning of the program, you can specify the number of moves (line 190) in which the player who is "it" must try to catch the other player. One player is a solid dark square, while the other is a light square with a border around it. The scores of the two players are constantly visible to the left and right of the "playing field"—a large rectangle surrounded by dark borders through which a player cannot pass.

One player moves up, down, left, or right, using the arrow keys (**E**, **X**, **S**, and **D**), while the other player uses the **P**, **.**, **L**, and **;** (P, period, L, and semicolon).

You can tell who is "it" by seeing which player has a square with a dot in the middle. Both players begin chasing or running until the one being chased gets caught (by occupying the same location as the player who is "it"), or until the maximum number of

```
470 IF FL$="L" THEN 520
480 CALL KEY(1,KEY,STATUS)
490 IF STATUS=0 THEN 520
500 FL$="L"
510 GOTO 550
520 CALL KEY(2,KEY,STATUS)
530 IF STATUS=0 THEN 480
540 FL$="R"
550 N=KEY
560 N$=STR$(N)
570 IF N=19 THEN 1510
580 IF N=02 THEN 680
590 IF N=03 THEN 710
600 IF N=05 THEN 740
610 IF N$="0" THEN 770
620 IF N=0 THEN 770
630 IF N=11 THEN 800
640 IF N=12 THEN 830
650 IF N=17 THEN 860
660 IF N=13 THEN 890
670 GOTO 450
680 CA=CA-1
690 GOSUB 1000
700 GOTO 450
710 CA=CA+1
720 GOSUB 1000
730 GOTO 450
740 RA=RA-1
750 GOSUB 1000
760 GOTO 450
770 RA=RA+1
780 GOSUB 1000
790 GOTO 450
800 RB=RB-1
810 GOSUB 1200
820 GOTO 450
830 CB=CB-1
840 GOSUB 1200
850 GOTO 450
```

(Continued in next column)

moves is reached. Then the appropriate player gets a point (for eluding or catching), the roles are switched, and the other player is "it."

Lines 220-290 define the squares (two with dots in the middle and two without dots). The row and column position of each player is kept track of in RA, CA, RB, and CB. Lines 480-520 check for one of the appropriate keys to be pressed, indicating in which direction a player's square is to be moved. These movements are effected in lines 680-900 and by the subroutines starting in line 1000 for one player and in line 1200 for the other. After each move, the positions are checked for boundaries that cannot be passed, for the squares occupying the same location, and for the maximum number of moves being reached.

Lines 470, 500, and 540 keep tabs on who moved last so that one player cannot monopolize all the moves simply because the part of the keyboard that player operates is scanned first.

Obviously, TAG is not the most advanced of arcade games, but it does show, in a fairly straightforward manner, how players can maneuver around a "board" and chase or run from an opponent.

(Continued from previous column)

```
860 CB=CB+1
870 GOSUB 1200
880 GOTO 450
890 RB=RB+1
```

```
 900 GOSUB 1200
 910 GOTO 450
 920 CALL CLEAR
 940 CALL CLEAR
 950 CALL HCHAR(1,5,96,24)
 960 CALL HCHAR(23,5,96,24)
 970 CALL VCHAR(2,5,96,22)
 980 CALL VCHAR(2,28,96,22)
 990 RETURN
1000 IF RA<2 THEN 1110
1010 IF RA>22 THEN 1110
1020 IF CA<6 THEN 1110
1030 IF CA>27 THEN 1110
1040 CALL VCHAR(RA2,CA2,32)
1050 RA2=RA
1060 CA2=CA
1070 CALL SOUND(100,392,3)
1080 IF DOT$="W" THEN 1100
1090 COUNT=COUNT+1
1100 GOTO 1130
1110 RA=RA2
1120 CA=CA2
1130 CALL VCHAR(RA,CA,96)
1140 IF RA<>RB THEN 1170
1150 IF CA<>CB THEN 1170
1160 ADD1$="A"
1170 IF DOT$="W" THEN 1190
1180 CALL VCHAR(RA,CA,98)
1190 RETURN
1200 IF RB<2 THEN 1310
1210 IF RB>22 THEN 1310
1220 IF CB<6 THEN 1310
1230 IF CB>27 THEN 1310
1240 CALL VCHAR(RB2,CB2,32)
1250 RB2=RB
1260 CB2=CB
1270 CALL SOUND(100,262,5)
1280 IF DOT$="B" THEN 1300
1290 COUNT=COUNT+1
```

(Continued in next column)

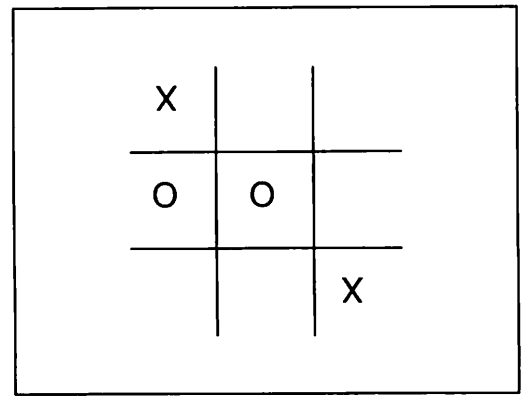
(Continued from previous column)

```
1300 GOTO 1330
1310 RB=RB2
1320 CB=CB2
1330 CALL VCHAR(RB,CB,97)
1340 IF RA<>RB THEN 1370
1350 IF CA<>CB THEN 1370
1360 ADD1$="A"
1370 IF DOT$="B" THEN 1390
1380 CALL VCHAR(RB,CB,99)
1390 RETURN
1400 CALL VCHAR(12,2,96,1)
1410 CALL VCHAR(12,30,97,1)
1420 SA$=STR$(SCOREA)
1430 SB$=STR$(SCOREB)
1440 CALL VCHAR(14,2,ASC(SEG$(SA$,1,1)),1)
1450 IF SCOREA<10 THEN 1470
1460 CALL VCHAR(14,3,ASC(SEG$(SA$,2,1)),1)
1470 CALL VCHAR(14,30,ASC(SEG$(SB$,1,1)),1)
1480 IF SCOREB<10 THEN 1500
1490 CALL VCHAR(14,31,ASC(SEG$(SB$,1,1)),1)
1500 RETURN
1510 CALL CLEAR
1520 END
```

TICHUMAN

```
100 REM * ************************
110 REM *        TICHUMAN        *
120 REM *                        *
130 REM *     COPYRIGHT 1983     *
140 REM *     DONNA S. PADGETT   *
150 REM * ************************
160 CALL CLEAR
170 WINP1=0
180 WINP2=0
190 NOWIN=0
200 MSG1$="H E L L O"
210 MSG2$="W E L C O M E"
220 MSG3$="T O"
230 MSG4$="T I C T A C T O E"
240 PRINT TAB(11);MSG1$
250 PRINT :::
260 PRINT TAB(9);MSG2$
270 PRINT :
280 PRINT TAB(14);MSG3$
290 PRINT :
300 PRINT TAB(6);MSG4$
310 PRINT ::
320 PRINT "X makes the first move."
330 INPUT "Player 1, are you X or O? ":P1$
340 IF P1$="X" THEN 380
350 IF P1$="O" THEN 400
360 PRINT "Please respond with X or O."
370 GOTO 330
380 P2$="O"
390 GOTO 410
400 P2$="X"
410 PRINT :
420 PRINT "Alright, Players, GOOD LUCK!!"
430 IF P1$="X" THEN 460
440 PRINT "PLAYER 2 GOES FIRST."
450 GOTO 470
460 PRINT "PLAYER 1 GOES FIRST."
```

TICHUMAN is a version of TICTACTO, the program described following this one. Unlike TICTACTO, TICHUMAN allows two people to play one another in tick-tack-toe without using any paper or bothering to keep score. In TICTACTO, however, you play against the TI-99/4A—which is pretty tough to beat!



Lines 170-190 set up the win counters to zero for both players, as well as the counter for ties. Then lines 200-350 print a welcome message to the screen and ask player 1, "Are you X or O?"

Since X always goes first, the program knows whose turn it is at any time. Lines 490-630 "paint" the tick-tack-toe grid on the screen. Enlarged "X" and "O" characters are defined by the **CALL CHAR** statements of lines 650-680.

Lines 690-740 clear the nine-element array MOVE$, which keeps track of the moves that have been made to the nine possible

```
470 PRINT :
480 INPUT "Press 'enter' to begin.":BEG$
490 REM *PUTS TICTACTO GRID ON SCREEN*
500 CALL CLEAR
510 CALL HCHAR(9,9,42,14)
520 CALL HCHAR(14,9,42,14)
530 CALL VCHAR(5,13,42,14)
540 CALL VCHAR(5,18,42,14)
550 CALL HCHAR(5,9,49)
560 CALL HCHAR(5,14,50)
570 CALL HCHAR(5,19,51)
580 CALL HCHAR(10,9,52)
590 CALL HCHAR(10,14,53)
600 CALL HCHAR(10,19,54)
610 CALL HCHAR(15,9,55)
620 CALL HCHAR(15,14,56)
630 CALL HCHAR(15,19,57)
640 REM *CREATES "X" AND "O"
    CHARACTERS*
650 X$="C3663C18183C66C3"
660 CALL CHAR(128,X$)
670 O$="183C66C3C3663C18"
680 CALL CHAR(129,O$)
690 REM *MOVE$ HOLDS MOVES MADE*
700 DIM MOVE$(9)
710 REM *S MEANS SPACE - NO VALUE*
720 FOR I=1 TO 9
730 MOVE$(I)="S"
740 NEXT I
750 RESTORE 800
760 FOR I=3 TO 13
770 READ A
780 CALL HCHAR(21,I,A)
790 NEXT I
800 DATA 80,76,65,89,69,82,32
810 DATA 49,32
820 DATA 61,32
830 CALL HCHAR(21,14,ASC(P1$))
840 REM*TURN$/P1=PLAYER1/P2=PLAYER2*
```

(Continued in next column)

locations on the grid. Lines 800-830 then remind you whether player 1 is X or O; with two players you might forget which you were and make a bad move.

Lines 850-1120 then determine whose move it is, spell out "Player ... Pick your Square," and wait for a number to be keyed. By keying a number between 1 and 9 in line 1130, the player attempts an appropriate move; if unsuccessful, the player must key a new try. The KEY statements of lines 1190-1290 analyze what number the player keyed, and lines 1300-1470 send the program to the correct square to record the move. If there is already an X or an O in that square, the player must key a move again (lines 1590-1600).

The X or O is moved to the appropriate position, and the board is checked for a win or a tie (lines 2240-2550). The turn is then given in lines 2560-2600 to the next player. If there is a win or a tie, the score is displayed, and another game can be played (lines 2610-2790).

(Continued from previous column)

```
850 IF P1$="X" THEN 880
860 TURN$="P2"
870 GOTO 890
880 TURN$="P1"
890 REM *NEXT LINES ASK PERSON FOR
    INPUT*
900 IF TURN$="P2" THEN 930
910 RESTORE 1040
920 GOTO 940
```

```
 930 RESTORE 1070
 940 FOR I=3 TO 12
 950 READ A
 960 CALL HCHAR(23,I,A)
 970 NEXT I
 980 RESTORE 1100
 990 FOR I=13 TO 29
1000 READ A
1010 CALL HCHAR(23,I,A)
1020 NEXT I
1030 REM *ASCII CODES FOR - PLAYER 1, *
1040 DATA 80,76,65,89,69,82,32
1050 DATA 49,44,32
1060 REM *ASCII CODES FOR- PLAYER 2, *
1070 DATA 80,76,65,89,69,82,32
1080 DATA 50,44,32
1090 REM *ASCII CODES FOR - "PICK YOUR
     SQUARE:"*
1100 DATA 80,73,67,75,32
1110 DATA 89,79,85,82,32
1120 DATA 83,81,85,65,82,69,58
1130 CALL KEY(0,KEY,STATUS)
1140 IF STATUS=0 THEN 1130
1150 REM *ERASES "PICK YOUR SQUARE"*
1160 CALL HCHAR(23,3,32,27)
1170 REM *ERASES "TRY AGAIN"*
1180 CALL HCHAR(24,7,32,10)
1190 IF KEY<49 THEN 1500
1200 IF KEY>57 THEN 1500
1210 IF KEY=49 THEN 1300
1220 IF KEY=50 THEN 1320
1230 IF KEY=51 THEN 1340
1240 IF KEY=52 THEN 1360
1250 IF KEY=53 THEN 1380
1260 IF KEY=54 THEN 1400
1270 IF KEY=55 THEN 1420
1280 IF KEY=56 THEN 1440
1290 IF KEY=57 THEN 1460
1300 J=1
```

(Continued in next column)

(Continued from previous column)

```
1310 GOTO 1590
1320 J=2
1330 GOTO 1590
1340 J=3
1350 GOTO 1590
1360 J=4
1370 GOTO 1590
1380 J=5
1390 GOTO 1590
1400 J=6
1410 GOTO 1590
1420 J=7
1430 GOTO 1590
1440 J=8
1450 GOTO 1590
1460 J=9
1470 GOTO 1590
1480 REM *BAD INPUT ERROR MSG HERE*
1490 REM *TRY AGAIN*
1500 RESTORE 1560
1510 FOR I=7 TO 15
1520 READ A
1530 CALL HCHAR(24,I,A)
1540 NEXT I
1550 REM *ASCII CODES FOR - TRY AGAIN*
1560 DATA 84,82,89,32
1570 DATA 65,71,65,73,78
1580 GOTO 1130
1590 IF MOVE$(J)="X" THEN 1500
1600 IF MOVE$(J)="O" THEN 1500
1610 IF TURN$="P2" THEN 1640
1620 MOVE$(J)=P1$
1630 GOTO 1660
1640 MOVE$(J)=P2$
1650 GOTO 1660
1660 REM *NEXT LINES MOVE IN APPROPRIATE
     X OR O*
1670 IF J=1 THEN 1780
1680 IF J=2 THEN 1810
```

```
1690 IF J=3 THEN 1840
1700 IF J=4 THEN 1870
1710 IF J=5 THEN 1900
1720 IF J=6 THEN 1930
1730 IF J=7 THEN 1960
1740 IF J=8 THEN 1990
1750 IF J=9 THEN 2020
1760 GOTO 2820
1770 REM *MOVE IN X'S HERE*
1780 IF MOVE$(J)="O" THEN 2060
1790 CALL HCHAR(7,11,128)
1800 GOTO 2240
1810 IF MOVE$(J)="O" THEN 2080
1820 CALL HCHAR(7,16,128)
1830 GOTO 2240
1840 IF MOVE$(J)="O" THEN 2100
1850 CALL HCHAR(7,21,128)
1860 GOTO 2240
1870 IF MOVE$(J)="O" THEN 2120
1880 CALL HCHAR(12,11,128)
1890 GOTO 2240
1900 IF MOVE$(J)="O" THEN 2140
1910 CALL HCHAR(12,16,128)
1920 GOTO 2240
1930 IF MOVE$(J)="O" THEN 2160
1940 CALL HCHAR(12,21,128)
1950 GOTO 2240
1960 IF MOVE$(J)="O" THEN 2180
1970 CALL HCHAR(17,11,128)
1980 GOTO 2240
1990 IF MOVE$(J)="O" THEN 2200
2000 CALL HCHAR(17,16,128)
2010 GOTO 2250
2020 IF MOVE$(J)="O" THEN 2220
2030 CALL HCHAR(17,21,128)
2040 GOTO 2240
2050 REM *MOVE IN O'S HERE*
2060 CALL HCHAR(7,11,129)
2070 GOTO 2240
```

(Continued in next column)

(Continued from previous column)

```
2080 CALL HCHAR(7,16,129)
2090 GOTO 2240
2100 CALL HCHAR(7,21,129)
2110 GOTO 2240
2120 CALL HCHAR(12,11,129)
2130 GOTO 2240
2140 CALL HCHAR(12,16,129)
2150 GOTO 2240
2160 CALL HCHAR(12,21,129)
2170 GOTO 2240
2180 CALL HCHAR(17,11,129)
2190 GOTO 2240
2200 CALL HCHAR(17,16,129)
2210 GOTO 2240
2220 CALL HCHAR(17,21,129)
2230 GOTO 2240
2240 REM *CHECKS FOR 3/ROW*
2250 ROW3$="N"
2260 DATA 1,2,3
2270 DATA 4,5,6
2280 DATA 7,8,9
2290 DATA 1,4,7
2300 DATA 2,5,8
2310 DATA 3,6,9
2320 DATA 1,5,9
2330 DATA 3,5,7
2340 RESTORE 2260
2350 FOR I=1 TO 8
2360 IF ROW3$="Y" THEN 2450
2370 READ K1,K2,K3
2380 IF MOVE$(K1)="S" THEN 2440
2390 IF MOVE$(K1)=MOVE$(K2)THEN 2410
2400 GOTO 2440
2410 IF MOVE$(K2)=MOVE$(K3)THEN 2430
2420 GOTO 2440
2430 ROW3$="Y"
2440 NEXT I
2450 REM *IF ROW3$="Y" - 3 IN A ROW!*
2460 IF ROW3$="Y" THEN 2610
```

```
2470 REM *CHECK FOR NO WIN*
2480 SPACE$="N"
2490 FOR I=1 TO 9
2500 IF SPACE$="Y" THEN 2550
2510 IF MOVE$(I)="X" THEN 2540
2520 IF MOVE$(I)="O" THEN 2540
2530 SPACE$="Y"
2540 NEXT I
2550 IF SPACE$="N" THEN 2690
2560 IF TURN$="P1" THEN 2590
2570 TURN$="P1"
2580 GOTO 890
2590 TURN$="P2"
2600 GOTO 890
2610 REM *IF HERE, 3/ROW*
2620 IF TURN$="P1" THEN 2660
2630 WINP2=WINP2+1
2640 PRINT "CONGRATULATIONS, PLAYER 2!!!"
2650 GOTO 2720
2660 WINP1=WINP1+1
2670 PRINT "CONGRATULATIONS, PLAYER 1!!!"
2680 GOTO 2720
2690 REM *IF HERE, CAT WON!*
2700 NOWIN=NOWIN+1
2710 PRINT "IT'S A DRAW!"
2720 PRINT :
2730 PRINT "THE SCORE IS: "
2740 PRINT "PLAYER 1: ";WINP1
2750 PRINT "PLAYER 2: ";WINP2
2760 PRINT "TIES: ";NOWIN
2770 PRINT :
2780 INPUT "DO YOU WANT TO PLAY AGAIN?
     (Y/N) ":YN$
2790 IF YN$="Y" THEN 320
2800 PRINT :
2810 PRINT "ALRIGHT, TILL NEXT TIME..."
2820 END
```

TICTACTO

```
100 REM * ************************
110 REM *                        *
120 REM *       TICTACTO         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONNA S. PADGETT     *
160 REM * ************************
170 CALL CLEAR
180 WINP=0
190 WINC=0
200 NOWIN=0
210 MSG1$="H E L L O"
220 MSG2$="W E L C O M E"
230 MSG3$="T O"
240 MSG4$="T I C T A C T O E"
250 PRINT TAB(11);MSG1$
260 PRINT::::
270 PRINT TAB(9);MSG2$
280 PRINT:
290 PRINT TAB(14);MSG3$
300 PRINT:
310 PRINT TAB(6);MSG4$
320 PRINT::
330 PRINT "X makes the first move."
340 INPUT "Do you want to be X or O? ":XO$
350 IF XO$="X" THEN 390
360 IF XO$="O" THEN 390
370 PRINT "Please respond with X or O."
380 GOTO 340
390 PRINT:
400 PRINT "Alright, ";XO$;", GOOD LUCK!!"
410 IF XO$="X" THEN 440
420 PRINT "I go first."
430 GOTO 450
440 PRINT "You go first."
450 PRINT:
460 INPUT "Press 'enter' to begin.":BEG$
```

TICTACTO, as mentioned earlier, is a game between a human and a computer, whereas TICHUMAN serves as an electronic blackboard and scoring machine for two people wanting to play tick-tack-toe.

TICTACTO has logic that is similar to that of TICHUMAN, but TICTACTO differs substantially. This program has sections in which the 99/4A has to figure out the best possible move in order to avoid getting caught in a no-win situation.

Lines 1410-3110 contain the logic that is significantly different. Lines 1420-1670 look for a win situation. (Why should the computer waste time looking for blocks or other moves if it can win right away?) Lines 1680-1920 look for situations in which the "human zapper" (the computer) must block in order to survive. Then lines 1930-2090 block the human player's corner between two sides to avoid a "double threat." Lines 2100-2140 will take the middle square if no better move is available. Finally, lines 2150-3110 look for progressively lower choice moves, always looking for corner moves since they offer good winning alternatives.

When a win or a tie is reached, you are asked if you want to play again, and the score to that point is given.

Good luck!

```
470 REM *PUTS TICTACTO GRID ON SCREEN*
480 CALL CLEAR
490 CALL HCHAR(9,9,42,14)
500 CALL HCHAR(14,9,42,14)
510 CALL VCHAR(5,13,42,14)
520 CALL VCHAR(5,18,42,14)
530 CALL HCHAR(5,9,49)
540 CALL HCHAR(5,14,50)
550 CALL HCHAR(5,19,51)
560 CALL HCHAR(10,9,52)
570 CALL HCHAR(10,14,53)
580 CALL HCHAR(10,19,54)
590 CALL HCHAR(15,9,55)
600 CALL HCHAR(15,14,56)
610 CALL HCHAR(15,19,57)
620 REM *CREATES "X" AND "O"
    CHARACTERS*
630 X$="C3663C18183C66C3"
640 CALL CHAR(128,X$)
650 O$="183C66C3C3663C18"
660 CALL CHAR(129,O$)
670 REM *MOVE$ HOLDS MOVES MADE*
680 DIM MOVE$(9)
690 REM *S MEANS SPACE - NO VALUE*
700 FOR I=1 TO 9
710 MOVE$(I)="S"
720 NEXT I
730 REM *TURN$/P=PERSON,C=COMPUTER*
740 IF XO$="X" THEN 780
750 TURN$="C"
760 COMP$="X"
770 GOTO 800
780 TURN$="P"
790 COMP$="O"
800 IF TURN$="C" THEN 1410
810 REM *NEXT LINES ASK FOR PERSON'S
    INPUT*
820 RESTORE 890
830 FOR I=7 TO 23
```

(Continued in next column)

(Continued from previous column)

```
840 READ A
850 CALL HCHAR(23,I,A)
860 NEXT I
870 REM *ASCII CODES FOR - PLAYER 1, *
880 REM *ASCII CODES FOR - PICK YOUR
    SQUARE: "*
890 DATA 80,73,67,75,32
900 DATA 89,79,85,82,32
910 DATA 83,81,85,65,82,69,58
920 CALL KEY(0,KEY,STATUS)
930 IF STATUS=0 THEN 920
940 REM *ERASES "PICK YOUR SQUARE"*
950 CALL HCHAR(23,7,32,17)
960 REM *ERASES "TRY AGAIN"*
970 CALL HCHAR(24,7,32,10)
980 IF KEY<49 THEN 1290
990 IF KEY>57 THEN 1290
1000 IF KEY=49 THEN 1090
1010 IF KEY=50 THEN 1110
1020 IF KEY=51 THEN 1130
1030 IF KEY=52 THEN 1150
1040 IF KEY=53 THEN 1170
1050 IF KEY=54 THEN 1190
1060 IF KEY=55 THEN 1210
1070 IF KEY=56 THEN 1230
1080 IF KEY=57 THEN 1250
1090 J=1
1100 GOTO 1370
1110 J=2
1120 GOTO 1370
1130 J=3
1140 GOTO 1370
1150 J=4
1160 GOTO 1370
1170 J=5
1180 GOTO 1370
1190 J=6
1200 GOTO 1370
1210 J=7
```

```
1220 GOTO 1370
1230 J=8
1240 GOTO 1370
1250 J=9
1260 GOTO 1370
1270 REM *BAD INPUT ERROR MSG HERE*
1280 REM *TRY AGAIN*
1290 RESTORE 1340
1300 FOR I=7 TO 15
1310 READ A
1320 CALL HCHAR(24,I,A)
1330 NEXT I
1340 DATA 84,82,89,32
1350 DATA 65,71,65,73,78
1360 GOTO 920
1370 IF MOVE$(J)="X" THEN 1290
1380 IF MOVE$(J)="O" THEN 1290
1390 MOVE$(J)=XO$
1400 GOTO 3120
1410 REM *NEXT LINES FIGURE COMPUTER'S
     MOVE*
1420 REM *FIRST LOOK FOR WIN*
1430 COMPMOVE=0
1440 RESTORE 3720
1450 FOR I=1 TO 8
1460 IF COMPMOVE>0 THEN 3100
1470 READ K1,K2,K3
1480 IF MOVE$(K1)=MOVE$(K2)THEN 1520
1490 IF MOVE$(K1)=MOVE$(K3)THEN 1540
1500 IF MOVE$(K2)=MOVE$(K3)THEN 1560
1510 GOTO 1660
1520 IF MOVE$(K1)=COMP$ THEN 1580
1530 GOTO 1660
1540 IF MOVE$(K1)=COMP$ THEN 1610
1550 GOTO 1660
1560 IF MOVE$(K2)=COMP$ THEN 1640
1570 GOTO 1660
1580 IF MOVE$(K3)=XO$ THEN 1660
1590 COMPMOVE=K3
```

(Continued in next column)

(Continued from previous column)

```
1600 GOTO 1660
1610 IF MOVE$(K2)=XO$ THEN 1660
1620 COMPMOVE=K2
1630 GOTO 1660
1640 IF MOVE$(K1)=XO$ THEN 1660
1650 COMPMOVE=K1
1660 NEXT I
1670 IF COMPMOVE>0 THEN 3100
1680 REM *2ND LOOK FOR BLOCK*
1690 RESTORE 3720
1700 FOR I=1 TO 8
1710 IF COMPMOVE>0 THEN 3100
1720 READ K1,K2,K3
1730 IF MOVE$(K1)=MOVE$(K2)THEN 1770
1740 IF MOVE$(K1)=MOVE$(K3)THEN 1790
1750 IF MOVE$(K2)=MOVE$(K3)THEN 1810
1760 GOTO 1910
1770 IF MOVE$(K1)=XO$ THEN 1830
1780 GOTO 1910
1790 IF MOVE$(K1)=XO$ THEN 1860
1800 GOTO 1910
1810 IF MOVE$(K2)=XO$ THEN 1890
1820 GOTO 1910
1830 IF MOVE$(K3)=COMP$ THEN 1910
1840 COMPMOVE=K3
1850 GOTO 1910
1860 IF MOVE$(K2)=COMP$ THEN 1910
1870 COMPMOVE=K2
1880 GOTO 1910
1890 IF MOVE$(K1)=COMP$ THEN 1910
1900 COMPMOVE>0 THEN 3100
1910 NEXT I
1920 IF COMPMOVE>0 THEN 3100
1930 REM *3RD BLOCK PERSONS CORNER
     BETWEEN TWO SIDES*
1940 RESTORE 2060
1950 FOR I=1 TO 4
1960 IF COMPMOVE>0 THEN 2140 IF
     COMPMOVE>0 THEN 3100
```

```
1970 READ K1,K2,K3
1980 IF MOVE$(K1)=XO$ THEN 2000
1990 GOTO 2050
2000 IF MOVE$(K2)=XO$ THEN 2020
2010 GOTO 2050
2020 IF MOVE$(K3)="S" THEN 2040
2030 GOTO 2050
2040 COMPMOVE=K3
2050 NEXT I
2060 DATA 2,4,1
2070 DATA 2,6,3
2080 DATA 6,8,9
2090 DATA 4,8,7
2100 REM *4TH TAKE MIDDLE SQUARE*
2110 IF MOVE$(5)="X" THEN 2140
2120 IF MOVE$(5)="O" THEN 2140
2130 COMPMOVE=5
2140 IF COMPMOVE>0 THEN 3100
2150 REM *5TH TRY FOR 3RD CORNER*
2160 RESTORE 2400
2170 FOR I=1 TO 4
2180 IF COMPMOVE>0 THEN 3100
2190 READ K1,K2,K3
2200 IF MOVE$(K1)=MOVE$(K2)THEN 2240
2210 IF MOVE$(K1)=MOVE$(K3)THEN 2290
2220 IF MOVE$(K2)=MOVE$(K3)THEN 2340
2230 GOTO 2390
2240 IF MOVE$(K1)=XO$ THEN 2390
2250 IF MOVE$(K3)=XO$ THEN 2390
2260 IF MOVE$(K3)=COMP$ THEN 2390
2270 COMPMOVE=K3
2280 GOTO 2390
2290 IF MOVE$(K1)=XO$ THEN 2390
2300 IF MOVE$(K2)=XO$ THEN 2390
2310 IF MOVE$(K2)=COMP$ THEN 2390
2320 COMPMOVE=K2
2330 GOTO 2390
2340 IF MOVE$(K2)=XO$ THEN 2390
2350 IF MOVE$(K1)=XO$ THEN 2390
```

(Continued in next column)

(Continued from previous column)

```
2360 IF MOVE$(K1)=COMP$ THEN 2390
2370 COMPMOVE=K1
2380 GOTO 2390
2390 NEXT I
2400 DATA 1,3,9
2410 DATA 1,7,9
2420 DATA 3,7,9
2430 DATA 1,3,7
2440 IF COMPMOVE>0 THEN 3100
2450 REM *6TH TRY FOR OPPOSITE CORNER*
2460 RESTORE 2470
2470 DATA 1,3,9
2480 DATA 1,7,9
2490 DATA 3,9,7
2500 DATA 3,1,7
2510 FOR I=1 TO 4
2520 IF COMPMOVE>0 THEN 3100
2530 READ K1,K2,K3
2540 IF MOVE$(K1)=XO$ THEN 2650
2550 IF MOVE$(K2)=XO$ THEN 2650
2560 IF MOVE$(K3)=XO$ THEN 2650
2570 IF MOVE$(K1)=COMP$ THEN 2600
2580 IF MOVE$(K3)=COMP$ THEN 2630
2590 GOTO 2650
2600 IF MOVE$(K3)=COMP$ THEN 2650
2610 COMPMOVE=K3
2620 GOTO 2650
2630 IF MOVE$(K1)=COMP$ THEN 2650
2640 COMPMOVE=K1
2650 NEXT I
2660 IF COMPMOVE>0 THEN 3100
2670 REM *7TH BLOCK PERSONS TRY FOR 3RD
     CORNER*
2680 RESTORE 2470
2690 FOR I=1 TO 4
2700 IF COMPMOVE>0 THEN 3100
2710 READ K1,K2,K3
2720 IF MOVE$(K1)=XO$ THEN 2740
2730 GOTO 2880
```

```
2740 IF MOVE$(K3)=XO$ THEN 2760
2750 GOTO 2880
2760 RESTORE 2890
2770 FOR N=1 TO 2
2780 READ K4,K5
2790 IF MOVE$(K4)=XO$ THEN 2860
2800 IF MOVE$(K5)=XO$ THEN 2860
2810 IF MOVE$(K4)=COMP$ THEN 2840
2820 COMPMOVE=K4
2830 GOTO 2880
2840 IF MOVE$(K5)=COMP$ THEN 2860
2850 COMPMOVE=K5
2860 GOTO 2880
2870 NEXT N
2880 NEXT I
2890 DATA 2,8
2900 DATA 4,6
2910 IF COMPMOVE>0 THEN 3100
2920 REM *8TH TRY FOR FREE CORNER*
2930 RESTORE 2940
2940 DATA 1,3,7,9
2950 FOR I=1 TO 4
2960 IF COMPMOVE>0 THEN 3100
2970 READ K1
2980 IF MOVE$(K1)=XO$ THEN 3010
2990 IF MOVE$(K1)=COMP$ THEN 3010
3000 COMPMOVE=K1
3010 NEXT I
3020 IF COMPMOVE>0 THEN 3100
3030 REM *9TH TAKE ANY AVAILABLE MOVE*
3040 FOR I=1 TO 9
3050 IF COMPMOVE>0 THEN 3100
3060 IF MOVE$(I)=XO$ THEN 3090
3070 IF MOVE$(I)=COMP$ THEN 3090
3080 COMPMOVE=I
3090 NEXT I
3100 J=COMPMOVE
3110 MOVE$(J)=COMP$
```

(Continued in next column)

(Continued from previous column)

```
3120 REM *NEXT LINES MOVE IN APPROPRIATE
     X OR O*
3130 IF J=1 THEN 3240
3140 IF J=2 THEN 3270
3150 IF J=3 THEN 3300
3160 IF J=4 THEN 3330
3170 IF J=5 THEN 3360
3180 IF J=6 THEN 3390
3190 IF J=7 THEN 3420
3200 IF J=8 THEN 3450
3210 IF J=9 THEN 3480
3220 GOTO 4280
3230 REM *MOVE IN X'S HERE*
3240 IF MOVE$(J)="O" THEN 3520
3250 CALL HCHAR(7,11,128)
3260 GOTO 3700
3270 IF MOVE$(J)="O" THEN 3540
3280 CALL HCHAR(7,16,128)
3290 GOTO 3700
3300 IF MOVE$(J)="O" THEN 3560
3310 CALL HCHAR(7,21,128)
3320 GOTO 3700
3330 IF MOVE$(J)="O" THEN 3580
3340 CALL HCHAR(12,11,128)
3350 GOTO 3700
3360 IF MOVE$(J)="O" THEN 3600
3370 CALL HCHAR(12,16,128)
3380 GOTO 3700
3390 IF MOVE$(J)="O" THEN 3620
3400 CALL HCHAR(12,21,128)
3410 GOTO 3700
3420 IF MOVE$(J)="O" THEN 3640
3430 CALL HCHAR(17,11,128)
3440 GOTO 3700
3450 IF MOVE$(J)="O" THEN 3660
3460 CALL HCHAR(17,16,128)
3470 GOTO 3710
3480 IF MOVE$(J)="O" THEN 3680
```

```
3490 CALL HCHAR(17,21,128)
3500 GOTO 3700
3510 REM *MOVE IN O'S HERE*
3520 CALL HCHAR(7,11,129)
3530 GOTO 3700
3540 CALL HCHAR(7,16,129)
3550 GOTO 3700
3560 CALL HCHAR(7,21,129)
3570 GOTO 3700
3580 CALL HCHAR(12,11,129)
3590 GOTO 3700
3600 CALL HCHAR(12,16,129)
3610 GOTO 3700
3620 CALL HCHAR(12,21,129)
3630 GOTO 3700
3640 CALL HCHAR(17,11,129)
3650 GOTO 3700
3660 CALL HCHAR(17,16,129)
3670 GOTO 3700
3680 CALL HCHAR(17,21,129)
3690 GOTO 3700
3700 REM *CHECKS FOR 3/ROW*
3710 ROW3$="N"
3720 DATA 1,2,3
3730 DATA 4,5,6
3740 DATA 7,8,9
3750 DATA 1,4,7
3760 DATA 2,5,8
3770 DATA 3,6,9
3780 DATA 1,5,9
3790 DATA 3,5,7
3800 RESTORE 3720
3810 FOR I=1 TO 8
3820 IF ROW3$="Y" THEN 3910
3830 READ K1,K2,K3
3840 IF MOVE$(K1)="S" THEN 3900
3850 IF MOVE$(K1)=MOVE$(K2)THEN 3870
3860 GOTO 3900
```

(Continued in next column)

(Continued from previous column)

```
3870 IF MOVE$(K2)=MOVE$(K3)THEN 3890
3880 GOTO 3900
3890 ROW3$="Y"
3900 NEXT I
3910 REM *IF ROW3$="Y" - 3 IN A ROW!*
3920 IF ROW3$="Y" THEN 4070
3930 REM *CHECK FOR NO WIN*
3940 SPACE$="N"
3950 FOR I=1 TO 9
3960 IF SPACE$="Y" THEN 4010
3970 IF MOVE$(I)="X" THEN 4000
3980 IF MOVE$(I)="O" THEN 4000
3990 SPACE$="Y"
4000 NEXT I
4010 IF SPACE$="N" THEN 4150
4020 IF TURN$="P" THEN 4050
4030 TURN$="P"
4040 GOTO 800
4050 TURN$="C"
4060 GOTO 800
4070 REM *IF HERE, 3/ROW*
4080 IF TURN$="P" THEN 4120
4090 WINC=WINC+1
4100 PRINT "SORRY, BUT YOU LOSE!"
4110 GOTO 4180
4120 WINP=WINP+1
4130 PRINT "CONGRATULATIONS, YOU WIN!"
4140 GOTO 4180
4150 REM *IF HERE, CAT WON!*
4160 NOWIN=NOWIN+1
4170 PRINT "IT'S A DRAW!"
4180 PRINT :
4190 PRINT "THE SCORE IS: "
4200 PRINT "COMPUTER: ";WINC
4210 PRINT "YOU: ";WINP
4220 PRINT "TIES: ";NOWIN
4230 PRINT :
```

```
4240 INPUT "DO YOU WANT TO PLAY AGAIN?
     (Y/N) ":YN$
4250 IF YN$="Y" THEN 330
4260 PRINT :
4270 PRINT "ALRIGHT, TILL NEXT TIME..."
4280 END
```

3

Business and Educational Programs

CKBOOK

```
100 REM * ************************
110 REM *          CKBOOK          *
120 REM *                          *
130 REM *       COPYRIGHT 1983     *
140 REM *     DONALD C. KREUTNER   *
150 REM *                          *
160 REM * ************************
170 DIM N2(100),A2(100),O2(100),N6(100),A6(100),
    O6(100),Q$(100)
180 REM READS IN A LIST OF CANCELLED
190 REM CHECKS AND CHECKS ISSUED
200 REM AND PRINTS A LIST OF CHECKS
210 DATA 110,"PUBLIC SERVICE INDIANA",24.82
220 DATA 118,"JOHNSON HARDWARE",20.00
230 DATA 200,"BAD CK NOT RECORDED",35.09
240 DATA 115,"COMPUTER MGT",35.73
250 DATA -9999,"END OF RETURNED
    CHECKS",999
260 DATA 110,24.83,111,25.00,113,17.82
270 DATA 115,35.73,117,16.57,118,20
280 DATA -999,999
290 FOR I=1 TO 100
300 READ N1,P$,A1
```

CKBOOK is a program that reads in a list of canceled checks and a list of issued checks, matches the two groups, and produces the following:

1. A list of returned checks, noting any discrepancies in amounts issued and paid
2. A message telling you whenever a check is paid that you failed to record as issued
3. A report of checks that are still outstanding and their amounts

The input for CKBOOK comes from the **DATA** statements in lines 210-280. Notice that these **DATA** statements are divided into two groups, one for returned checks and one for issued checks. In lines 210 to 250, input for returned checks includes the check number, a comma, the payee in quotes, a comma, and the check amount.

```
310 IF N1=-9999 THEN 390
320 N2(I)=N1
330 Q$(I)=P$
340 A2(I)=A1
350 C2=I
360 O2(I)=I
370 NEXT I
380 LCTR=0
390 REM SORT THE CKS RETURNED
400 REM ********************
410 INPUT "PRINTER (Y/N): ":PYN$
420 IF PYN$<>"Y" THEN 460
430 OPEN #1:"RS232.BA=1200"
440 PRINT #1:" CK#";TAB(10);"PAYEE";TAB(40);
    "AMT RETURNED";TAB(55);"AMT ISSUED"
450 PRINT #1
460 CALL CLEAR
470 S=0
480 FOR I=1 TO C2-1
490 IF N2(I)>N2(I+1)THEN 510
500 GOTO 580
510 T=O2(I)
520 T1=N2(I)
530 O2(I)=O2(I+1)
540 N2(I)=N2(I+1)
550 O2(I+1)=T
560 N2(I+1)=T1
570 S=1
580 NEXT I
590 IF S=1 THEN 470
600 REM READ IN ISSUED CKS
610 REM ********************
620 FOR I=1 TO 100
630 READ N5,A5
640 IF N5=-999 THEN 700
650 N6(I)=N5
660 A6(I)=A5
670 C6=I
680 O6(I)=I
690 NEXT I
```

The returned checks are terminated by a -9999 for the check number. (See line 250.)

Input for the issued checks includes check number, a comma, the amount of the check, a comma, the next check number, a comma, the amount for that check, etc. The end of the issued checks is designated by a -999 check number. (See line 280.) You can add to, alter, and delete these **DATA** statements to balance your own checkbook.

Lines 290-370 read the returned checks into several arrays (N2 for the check number, Q$ for the payee, and A2 for the amount). When the -9999 check number is encountered in line 310, a sort of the data for the returned checks begins in line 470 and continues to line 590. In line 410 you are asked if you want a printout.

Then the data for the issued checks is read in and sorted, in similar fashion, in lines 600-820.

The returned check list is printed in lines 850-1220. The check numbers of checks returned are compared, one at a time, to check numbers of the checks issued, in order to find a match (lines 960-1050). If a match is found, the amounts of those two checks are compared. If they are not equal, the discrepancy is clearly noted on the listing, and the totals of issued and returned checks will be different.

Then in lines 1280-1520, an outstanding check list is given, telling you which checks have not been returned, their amounts, and the total amount still outstanding. Lines 1530-1720 and lines 1730-1820 contain subroutines used to print details in portions

```
700 S=0
710 FOR I=1 TO C6-1
720 IF N6(I)>N6(I+1)THEN 740
730 GOTO 810
740 T=O6(I)
750 T1=N6(I)
760 O6(I)=O6(I+1)
770 N6(I)=N6(I+1)
780 O6(I+1)=T
790 N6(I+1)=T1
800 S=1
810 NEXT I
820 IF S=1 THEN 700
830 A7=0
840 A8=0
850 REM PRINT THE RETURNED CKS
860 REM ********************
870 PRINT "RETURNED CHECK LIST"
880 PRINT "-------------------"
890 IF PYN$<>"Y" THEN 940
900 PRINT #1:"*******************************
    ************************************"
910 PRINT #1:"*CHECKS RETURNED*"
920 PRINT #1:"*******************************
    ************************************"
930 PRINT #1
940 LCTR=LCTR+2
950 FOR I=1 TO C2
960 FOR J=1 TO C6
970 IF N2(I)=N6(J)THEN 1070
980 NEXT J
990 GOSUB 1530
1000 PRINT
1010 PRINT "NO MATCH FOR: ";N2(I)
1020 IF PYN$<>"Y" THEN 1050
1030 PRINT #1:"* NO MATCH FOR CK#:";N2(I);"*"
1040 PRINT #1
1050 LCTR=LCTR+2
1060 GOTO 1090
```

(Continued in next column)

of the program that deal with returned and outstanding checks.

(Continued from previous column)

```
1070 GOSUB 1530
1080 A8=A8+A6(J)
1090 NEXT I
1100 REMA7=TOT RETURNED
1110 REMA8=TOT ISSUED
1120 FOR I=1 TO C2
1130 A7=A7+A2(I)
1140 NEXT I
1150 REM PRINT THE OUTSTANDING CKS
1160 REM ********************
1170 PRINT
1180 PRINT "**************************"
1190 PRINT "TOTAL ISSUED: ";A8
1200 PRINT "TOTAL RETURNED:";A7
1210 IF PYN$<>"Y" THEN 1230
1220 PRINT #1:"*TOTALS*";TAB(40);A7;TAB(55);A8
1230 LCTR=LCTR+4
1240 IF LCTR<20 THEN 1270
1250 INPUT " ":CO$
1260 LCTR=0
1270 A7=0
1280 REM A7 NOW = TOT OUTSTANDING CKS
1290 REM ********************
1300 PRINT
1310 PRINT "OUTSTANDING CHECK LIST"
1320 PRINT "-------------------"
1330 LCTR=LCTR+3
1340 IF PYN$<>"Y" THEN 1390
1350 PRINT #1
1360 PRINT #1:"*******************************
     ************************************"
```

```
1370 PRINT #1:"*OUTSTANDING CHECKS*"
1380 PRINT #1:"*****************************
     ***********************************"
1390 FOR I=1 TO C6
1400 FOR J=1 TO C2
1410 IF N6(I)=N2(J)THEN 1450
1420 NEXT J
1430 A7=A7+A6(O6(I))
1440 GOSUB 1730
1450 NEXT I
1460 PRINT
1470 PRINT "TOTAL OUTSTANDING: ";A7
1480 IF PYN$<>"Y" THEN 1510
1490 PRINT #1
1500 PRINT #1:"*TOTAL OUTSTANDING*";
     TAB(55);A7
1510 LCTR=LCTR+2
1520 GOTO 1830
1530 PRINT
1540 PRINT "CK# ";N6(J);"FOR";A6(O6(J))
1550 PRINT "RETURNED FROM ";Q$(O2(I));
     A2(O2(I))
1560 IF PYN$<>"Y" THEN 1580
1570 PRINT #1:N6(J);TAB(10);Q$(O2(I));TAB(40);
     A2(O2(I));TAB(55);A6(O6(J))
1580 LCTR=LCTR+4
1590 IF LCTR<20 THEN 1620
1600 INPUT " ":CO$
1610 LCTR=0
1620 IF A2(O2(I))=A6(O6(J))THEN 1720
1630 PRINT "***RETURNED AMT NOT =
     REGISTER AMT***"
1640 IF PYN$<>"Y" THEN 1680
1650 PRINT #1
1660 PRINT #1:"*RETURNED AMT NOT =
     REGISTER AMT*"
1670 PRINT #1
1680 LCTR=LCTR+2
1690 IF LCTR<20 THEN 1720
```

(Continued in next column)

(Continued from previous column)

```
1700 INPUT " ":CO$
1710 LCTR=0
1720 RETURN
1730 PRINT
1740 PRINT "CK#";N6(I);"FOR";A6(O6(I))
1750 PRINT "STILL OUTSTANDING"
1760 IF PYN$<>"Y" THEN 1780
1770 PRINT #1:N6(I);TAB(55);A6(O6(I))
1780 LCTR=LCTR+3
1790 IF LCTR<20 THEN 1820
1800 INPUT " ":CO$
1810 LCTR=0
1820 RETURN
1830 IF PYN$<>"Y" THEN 1850
1840 CLOSE #1
1850 END
```

```
 CK#     PAYEE                              AMT RETURNED    AMT ISSUED

**********************************************************************
   *CHECKS RETURNED*
**********************************************************************

 110     PUBLIC SERVICE INDIANA             24.82           24.83

   *RETURNED AMT NOT = REGISTER AMT*

 115     COMPUTER MGT                       35.73           35.73
 118     JOHNSON HARDWARE                   20              20
 0       BAD CK NOT RECORDED                35.09           0

   *RETURNED AMT NOT = REGISTER AMT*

   * NO MATCH FOR CK#: 200 *

   *TOTALS*                                 115.64          80.56

**********************************************************************
   *OUTSTANDING CHECKS*
**********************************************************************
 111                                                       25
 113                                                       17.82
 117                                                       16.57

   *TOTAL OUTSTANDING*                                     59.39
```

EXPENSE

```
100 REM * ************************
110 REM *                        *
120 REM *        EXPENSE         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM *                        *
170 REM * ************************
180 OPTION BASE 1
190 DIM EXP$(200)
200 DIM N$(12)
210 REM MM/DD/YY ####### DESCRIPTION
220 CALL CLEAR
230 PRINT "      ***EXPENSE REPORT***"
240 PRINT
250 INPUT "CASSETTE OR DISK (C/D): ":CD$
260 IF CD$="C" THEN 300
270 INPUT "DSK1.FILENAME: ":D$
280 OPEN #1:D$
290 GOTO 310
300 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
310 FOR I=1 TO 200
320 INPUT #1:EXP$(I)
330 IF EXP$(I)="END" THEN 360
340 PRINT I
350 NEXT I
360 CTRN=I-1
370 CLOSE #1
380 REM ********************
390 CALL CLEAR
400 PRINT "P    PRINTER"
410 PRINT "L    LIST TO SCREEN"
420 PRINT "E    END PROGRAM"
430 PRINT
440 INPUT "SELECTION: ":S$
450 CALL CLEAR
460 TOT=0
```

EXPENSE is designed to provide you with a list of expense items and a total of their amounts. This program is ideal for month-end expense summaries for salesmen and others who need to keep track of business expenses.

Once again, the input for the program is created by one of the two EDIT programs, EDIT1 or EDIT2. EDIT2 is perhaps the most flexible, since it allows you to resequence your data and insert records where you want them, keeping them properly sequenced by month, day, and year.

The data you key must be in the following format:

Pos. 1-8	Date, by month/day/year (MM/DD/YY)
Pos. 10-16	Amount of the expense, with numbers of exactly seven digits Example: $45.00 is keyed as 0004500
Pos. 18 to end of record	Description of the expense

Following are examples of input data:

```
09/01/83 0004500 AUTO
09/05/83 0025000 RENT EXP
09/10/83 0000100 PARKING
09/13/83 0050000 SALARIES
09/14/83 0002000 DUES
```

Notice that all the date fields and the amount fields must be of the same length, since the program "expects" to find those

```
470 IF S$="E" THEN 1110
480 IF S$="L" THEN 570
490 IF S$<>"P" THEN 390
500 REM ********************
510 OPEN #2:"RS232.BA=1200"
520 INPUT "ENTER MM/DD/YY: ":DATE$
530 PRINT #2:TAB(20);"EXPENSE REPORT - ";
    DATE$
540 PRINT #2
550 PRINT #2:"DATE";TAB(20);"DESCRIPTION";
    TAB(57);"AMOUNT"
560 PRINT #2
570 FOR J=1 TO CTRN
580 IF EXP$(J)="END" THEN 750
590 DT$=SEG$(EXP$(J),1,8)
600 AMT$=SEG$(EXP$(J),10,7)
610 DESC$=SEG$(EXP$(J),18,LEN(EXP$(J))-17)
620 AMT=VAL(AMT$)
630 TOT=TOT+AMT
640 NUM1=AMT
650 GOSUB 870
660 PRINT
670 PRINT DT$;TAB(15);NUM2$
680 PRINT DESC$
690 PRINT "**************************"
700 IF S$="P" THEN 730
710 INPUT " ":X$
720 IF S$<>"P" THEN 740
730 PRINT #2:DT$;TAB(20);DESC$;TAB(50);NUM2$
740 NEXT J
750 PRINT
760 NUM1=TOT
770 GOSUB 870
780 PRINT "**TOTAL EXPENSES** ";NUM2$
790 IF S$="P" THEN 820
800 INPUT " ":X$
810 IF S$<>"P" THEN 390
820 PRINT #2
830 PRINT #2:"**TOTAL EXPENSES**";TAB(50);
    NUM2$
```

fields in those exact locations. However, the description fields above vary from four to eight characters in length. They could be much longer (for example, 30 to 40 characters, or even more).

Line 250 asks you whether the data file to be used for input is on tape or disk. After the file is loaded into memory by lines 310-370, you are given the option of listing the report to the screen (**L**) or to the printer (**P**), or of ending the program (**E**) in lines 390-440.

If you choose to list the report to the printer, line 520 first asks for the date you want on the heading of the printout. As described earlier, three main details are given on the report: the date, the amount, and the description.

The amount is edited from the raw input data, using the subroutine in lines 860-1100. For instance, 0002000 is changed to 20.00. You may notice that this routine differs from other similar routines that are in the book. This difference simply illustrates that in programming, there are a multitude of alternatives. Even though a particular programmer designs a program one way, the design may not be perfect or even the best. It may not even be good. If you let your imagination flow while writing programs, you will frequently find better alternatives to those you've used before.

In this subroutine, NUM1$ contains the number to be edited. The 12 parts of the array N$ are set to space in lines 890-910. Then the number is examined, right to left, and a decimal point is inserted in the tenth position (lines 920 and 980-990). Finally, NUM2$ (the edited result) is created by

```
840 CLOSE #2
850 GOTO 390
860 REM ********************
870 NUM1$=STR$(NUM1)
880 LENN=LEN(NUM1$)
890 FOR I=1 TO 12
900 N$(I)=" "
910 NEXT I
920 N$(10)="."
930 CTR=13
940 FOR I=LENN TO 1 STEP -1
950 CTR=CTR-1
960 IF CTR<1 THEN 1020
970 POS$=SEG$(NUM1$,I,1)
980 IF CTR<>10 THEN 1000
990 CTR=CTR-1
1000 N$(CTR)=POS$
1010 NEXT I
1020 IF N$(11)<>" " THEN 1040
```

(Continued in next column)

concatenating (joining) the 12 parts of N$ in lines 1070-1090.

(Continued from previous column)

```
1030 N$(11)="0"
1040 IF N$(12)<>" " THEN 1060
1050 N$(12)="0"
1060 NUM2$=" "
1070 FOR I=1 TO 12
1080 NUM2$=NUM2$&N$(I)
1090 NEXT I
1100 RETURN
1110 END
```

```
                EXPENSE REPORT - 09/30/83

  DATE          DESCRIPTION                        AMOUNT

09/01/83        TRIP EXPENSE                        40.50
09/02/83        MISCELLANEOUS                       50.00
09/02/83        PARKING                              1.50
09/02/83        SUPPLIES                           500.98
09/05/83        MISC EXPENSES                       35.00
09/10/83        OFFICE SALARY                      200.00
09/15/83        CHAIR                              100.00
09/20/83        MILEAGE EXPENSE                    198.75
09/23/83        PAPER                                3.00
09/25/83        MILEAGE EXPENSE                    232.43
09/30/83        ENTERTAINMENT                       54.30

**TOTAL EXPENSES**                                1416.46
```

FLASHCARD

```
100 REM * ************************
110 REM *                        *
120 REM *        FLASHCARD       *
130 REM *                        *
140 REM *      COPYRIGHT 1983    *
150 REM *    DONALD C. KREUTNER  *
160 REM * ************************
170 OPTION BASE 1
180 DIM Q$(200)
190 DIM A$(200)
200 E=0
210 C=0
220 LAST1=0
230 RANDOMIZE
240 CALL CLEAR
250 PRINT "       FLASHCARD"
260 PRINT
270 PRINT "C    CLEAR PREVIOUS QUIZ"
280 PRINT "R    READ QUIZ FROM FILE"
290 PRINT "Q    QUIZ WITH QUESTIONS"
300 PRINT "A    QUIZ WITH ANSWERS"
310 PRINT "E    END PROGRAM"
320 PRINT
330 INPUT "SELECTION: ":S$
340 IF S$="C" THEN 380
350 IF S$="R" THEN 410
360 IF S$="Q" THEN 630
370 IF S$="A" THEN 630
380 IF S$="E" THEN 1000
390 LAST1=0
400 GOTO 240
410 INPUT "CASSETTE OR DISK (C/D): ":CD$
420 IF CD$="C" THEN 470
430 IF CD$<>"D" THEN 410
440 INPUT "DSK1.FILENAME: ":D$
450 OPEN #1:D$
460 GOTO 480
```

FLASHCARD is a program that can be of immense benefit to any student.

Many programs may be purchased to help you learn about different subjects. With most of these programs, however, you are forced to use the data that comes with the programs. Usually, you are not able to influence the selection of questions and answers on which you want to be drilled.

But with FLASHCARD you can key in all the new Spanish words, for example, as you encounter them, chapter by chapter. Thus, you can tailor data files for your own personal use, rather than review information you may not even need in the actual course work you are taking.

You can also save practically unlimited amounts of information on cassette or disk files for use at any time. Thus, you can effectively collect, add to, and review all the educational data you need.

What a tool this program can be for any subject! Students can drill on their special problem areas at their own pace and receive individual, "computerized" attention with the benefits of immediate feedback.

The beauty of this simple program is that you can ask any kind of question on any subject. The input files for FLASHCARD, as for many other programs in this book, can be created and modified easily by using EDIT1 or EDIT2. The first record the program reads will be considered the "ques-

```
470 OPEN #1:"CS1",INTERNAL,FIXED 128,INPUT
480 IF LAST1<200 THEN 510
490 PRINT "200 MAXIMUM QUESTIONS"
500 GOTO 240
510 FOR I=LAST1+1 TO 200
520 INPUT #1:X$
530 IF X$="END" THEN 610
540 PRINT I
550 Q$(I)=X$
560 INPUT #1:X$
570 A$(I)=X$
580 IF X$="END" THEN 610
590 LAST1=I
600 NEXT I
610 CLOSE #1
620 GOTO 240
630 CALL CLEAR
640 INPUT "SEQUENTIAL OR RANDOM
    (S/R): ":SR$
650 R=0
660 IF SR$="R" THEN 710
670 R=R+1
680 IF R<=LAST1 THEN 720
690 R=1
700 GOTO 720
710 R=INT(LAST1*RND)+1
720 IF S$="A" THEN 750
730 PRINT Q$(R)
740 GOTO 760
750 PRINT A$(R)
760 INPUT A1$
770 IF A1$="END" THEN 240
780 IF LEN(A1$)<1 THEN 240
790 IF A1$="STOP" THEN 240
800 IF S$="Q" THEN 830
810 IF A1$=Q$(R)THEN 940
820 GOTO 840
830 IF A1$=A$(R)THEN 940
840 PRINT "*****"
850 PRINT "WRONG"
```

tion" and the next record the "answer"; the third record will be another question followed by another answer; and so on, until the end of the file. Up to 200 questions and answers can be read in by FLASHCARD for use at one time. But the length of the questions and answers can limit how many items can be maintained in memory.

The arrays of Q$ and A$ hold the questions and answers (lines 180-190). C and E keep track of how many correct and incorrect answers the person who is answering has obtained since the beginning of the program. And LAST1 (line 220) "remembers" where the last question/answer is located within the Q$ and A$ arrays.

You can clear the memory of a quiz that you've read in, or you can accumulate one or more files by not clearing memory (using the command **C**). You can read in a file (**R**), and you are allowed two modes of quizzes. **Q** means you want the standard question-and-answer quiz, whereas **A** means you want to be "asked" the answers and respond with the questions.

A good way to use this last feature (in Spanish, for example) would be to give either the English answers or the Spanish answers simply by asking for the questions or answers (**Q** or **A**). If you consider home computer prices, this program alone makes the computer a worthwhile investment for not only a school but also a family.

In FLASHCARD, when you ask for a quiz, you can specify whether you want the questions to be asked randomly or whether you want to go through the questions one at a time, repetitively. The random approach

```
 860 PRINT
 870 PRINT Q$(R)
 880 PRINT A$(R)
 890 PRINT "*****"
 900 E=E+1
 910 PRINT
 920 PRINT "ERRORS =";E;" CORRECT =";C
 930 GOTO 660
 940 PRINT
 945 PRINT Q$(R)
 950 PRINT A$(R)
 960 C=C+1
 970 PRINT "ERRORS =";E;" CORRECT =";C
 980 PRINT
 990 GOTO 660
1000 CALL CLEAR
1010 PRINT "CORRECT:";C
1020 PRINT "ERRORS:";E
1030 IF E+C<1 THEN 1050
1040 PRINT INT((100*C)/(E+C));"%"
1050 END
```

helps you to master the material so that you are not dependent on the order of the questions. At times, especially when you are reviewing information, you may prefer to go through the list of questions (or answers) quickly, over and over again. As one professor used to say, "Repetition and review, repetition and review." The key to learning is proper training.

At the end of the program, the number correct and the number of errors are displayed, along with the percentage of correct answers.

LA PLUMA	DAR	USTED	EL DIA
THE PEN	TO GIVE	YOU	THE DAY
LA MUJER	EL COLOR	ELLA	LUNES
THE WOMAN	THE COLOR	SHE	MONDAY
EL HOMBRE	EL TREN	EL	MARTES
THE MAN	THE TRAIN	HE	TUESDAY
EL MUCHACHO	EL AVION	HABLAR	MIERCOLES
THE BOY	THE AIRPLANE	TO SPEAK	WEDNESDAY
EL HERMANO	LA MANANA	INGLES	JUEVES
THE BROTHER	THE MORNING	ENGLISH	THURSDAY
LA MUCHACHA	LA TARDE	ESPANOL	VIERNES
THE GIRL	THE AFTERNOON	SPANISH	FRIDAY
LA HERMANA	BUENAS NOCHES	QUIEN	SABADO
THE SISTER	GOOD EVENING	WHO	SATURDAY
YO	LA NOCHE	EL DINERO	DOMINGO
I	THE NIGHT	THE MONEY	SUNDAY

GENLED1

```
100 REM ************************
110 REM *                      *
120 REM *       GENLED1        *
130 REM *                      *
140 REM *    COPYRIGHT 1983    *
150 REM *  DONALD C. KREUTNER  *
160 REM ************************
170 OPTION BASE 1
180 DIM A$(500)
190 DIM A2$(500)
200 FOR I=1 TO 500
210 A$(I)="@"
220 NEXT I
230 LAST1=1
240 REM GENLED1 IS A MODIFIED VERSION
250 REM OF EDIT2 THAT ALLOWS YOU
260 REM TO KEY FIELD BY FIELD THE
270 REM INPUT DATA FOR GENLED2
280 REM TO AVOID DATA ENTRY ERRORS
290 REM *****COMMANDS*****
300 GOTO 440
310 INPUT "/":B$
320 IF B$="HELP" THEN 480
330 IF B$="C" THEN 200
340 IF B$="A" THEN 620
350 IF B$="D" THEN 1010
360 IF B$="T" THEN 1150
370 IF B$="L" THEN 1340
380 IF B$="K" THEN 1410
390 IF B$="LU" THEN 1560
400 IF B$="R" THEN 1630
410 IF B$="P" THEN 1830
420 IF B$="PU" THEN 1930
430 IF B$="E" THEN 2010
440 REM *****HELP SCREEN*****
450 CALL CLEAR
460 PRINT "*GENLED1*"
```

The following two programs, GENLED1 and GENLED2, are to be used together. GENLED1 is used to key financial data, whereas GENLED2 reads that data and produces a formatted report with totals. They accumulate income and/or expense data in a disk or cassette file and produce reports summarizing the monthly and year-to-date totals of all transactions by account number and date. Together, these programs can be a most versatile tool for a small business in maintaining documents like general ledgers or for home use in tracking income and expense information. You are free to set up your own "chart of accounts," and all the data files created by GENLED1 can be added to or changed as new transactions become available. Note that in the following descriptions of each program, frequent reference is made to the other program.

GENLED1 is actually a modified version of EDIT2. In fact, you can use either EDIT1 or EDIT2 to key the data for GENLED2. However, the advantage to a program like GENLED1 is that some editing of the fields is done as they are entered. For example, the date is checked for slashes in the 3rd and 5th positions of the eight-character date field (lines 740-760). The source code (check number or any other identifying information for each transaction) must be exactly four characters long (lines 790-910). The account number and payee number must each be exactly three characters long (line 700 and 780). These data checks

```
470 PRINT
480 PRINT "HELP PRINT INSTRUCTS"
490 PRINT "A     ADD LINES"
500 PRINT "C     CLEAR WORKFILE"
510 PRINT "D     DELETE LINES"
520 PRINT "T     TEXT IN A FILE"
530 PRINT "L     LIST WORKFILE"
540 PRINT "K     KEEP A FILE"
550 PRINT "LU    LIST UNNUMBERED"
560 PRINT "R     RENUMBER WORKFILE"
570 PRINT "P     PRINT WORKFILE"
580 PRINT "PU    PRINT UNNUMBERED"
590 PRINT "E     END PROGRAM"
600 GOTO 310
610 REM ***ADD LINES***
620 INPUT "LINE#: ":A
630 INPUT "STEP#: ":STEP1
640 IF STEP1<>0 THEN 660
650 STEP1=5
660 IF A$(A)<>"@" THEN 970
670 PRINT "REC# ";A
680 INPUT "ACCT# (OR END):":ACCT$
690 IF A<LAST1 THEN 730
700 IF LEN(ACCT$)<>3 THEN 680
710 IF A<LAST1 THEN 730
720 LAST1=A
730 IF ACCT$="END" THEN 990
740 INPUT "DATE (MM/DD/YY): ":DATE$
750 IF SEG$(DATE$,3,1)<>"/" THEN 740
760 IF SEG$(DATE$,6,1)<>"/" THEN 740
770 INPUT "PAYEE#: ":PAY$
780 IF LEN(PAY$)<>3 THEN 770
790 INPUT "SOURCE: ":SRC$
800 IF LEN(SRC$)=4 THEN 920
810 IF LEN(SRC$)>4 THEN 790
820 IF LEN(SRC$)<>3 THEN 850
830 SRC$=SRC$&" "
840 GOTO 920
850 IF LEN(SRC$)<>2 THEN 880
860 SRC$=SRC$&"  "
```

greatly reduce the possibility of keying errors.

GENLED1 can be modified to create data for other programs by altering the lines that do the actual keying and editing (lines 619-990). You can key any fields and join them together into a record, as in line 940.

All data to be used by GENLED2 is contained in one string of character data; that is, there are no integers or real numbers in the file itself. The first three characters represent the account number of the transaction. Positions 4-11 contain the eight-digit transaction date separated by two slashes. Note that each digit must be keyed. (For example, January 1, 1983, is keyed as **01/01/83.**) Positions 12-14 contain the three-digit payee number. (You must decide what numbers you want to assign to each payee, just as you must create your own account number descriptions in the DATA statements of GENLED2.)

Notice that if an account is an income account, the "payee" can be used to classify different categories of the same account number and to keep appropriate subtotal information. The source field in positions 15-18 is used to record the check number or other appropriate identifying information for the transaction. For example, check #101 could have 101 or 0101 in the position 4 source field. The amount of the transaction is encoded starting in position 19. Leading zeros are suppressed from the amount field. The number of account numbers and payee numbers allowed is set at 100 each by SIZEA.

```
 870 GOTO 920
 880 IF LEN(SRC$)<>1 THEN 910
 890 SRC$=SRC$&"  "
 900 GOTO 920
 910 SRC$="  "
 920 INPUT "AMOUNT: ":AMT
 930 AMT$=STR$(AMT)
 940 A$(A)=ACCT$&DATE$&PAY$&SRC$&AMT$
 950 A=A+STEP1
 960 GOTO 660
 970 PRINT "THIS LINE ALREADY EXISTS"
 980 PRINT A$(A)
 990 GOTO 310
1000 REM *****DELETE LINES*****
1010 INPUT "BEG LINE#: ":B
1020 INPUT "END LINE#: ":E
1030 IF E<LAST1 THEN 1060
1040 LAST1=B
1050 A$(B)="END"
1060 IF E>500 THEN 1020
1070 FOR I=B TO E
1080 IF A$(I)="@" THEN 1120
1090 REM ***SAME AS ABOVE LINES***
1100 PRINT " ";A$(I)
1110 A$(I)="@"
1120 NEXT I
1130 GOTO 310
1140 REM *****TEXT IN FILE*****
1150 INPUT "CASSETTE/DISK (C/D): ":CD$
1160 IF CD$="D" THEN 1200
1170 IF CD$<>"C" THEN 1150
1180 OPEN #1:"CS1",INPUT ,INTERNAL,FIXED 128
1190 GOTO 1220
1200 INPUT "INPUT FILE NAME: ":F$
1210 OPEN #1:F$
1220 INPUT "STEP#: ":STEP1
1230 IF STEP1<>0 THEN 1250
1240 STEP1=5
1250 FOR I=LAST1 TO 500 STEP STEP1
1260 INPUT #1:A$(I)
```

These limits can be changed by modifying the statements in lines 210-240 of GENLED2, but raising them too high can cause the report program to run out of memory. For most applications fewer than 100 of each should be quite satisfactory. Obviously, this tool is not designed for a very large business (although appropriate summary financial information can be provided).

The **DATA** statements in lines 330-680 in GENLED2 provide the account names and payee information for the desired numbers. As mentioned earlier, these lines can be changed for your own personal use. Lines 780-880 read in the account and payee data and provide a printout, if one is desired. Lines 910-1000 cause a data file to be read in from cassette or diskette and print a report (to the screen only or to a printer) as the file is read.

Because the report appears simultaneously with the file that is read, you must maintain the records in the proper order; that is, all of the records for account number 001 should be first, in the order of date sequence. With this method, month-to-date and year-to-date totals can be obtained from the file for any month.

The data entry portion of GENLED1 has several interesting features to ensure that only "clean" data is entered. Of course, more edits can be added for any other errors you may want to check.

If you are starting a new data file, you can begin with record number 1 after responding **A** to the beginning menu screen. You are required to give the starting record

```
1270 IF A$(I)="END" THEN 1300
1280 LAST1=I
1290 NEXT I
1300 CLOSE #1
1310 A$(I)="@"
1320 GOTO 310
1330 REM *****LIST WORKFILE*****
1340 FOR I=1 TO 500
1350 IF I>LAST1 THEN 1390
1360 IF A$(I)="@" THEN 1380
1370 PRINT I;A$(I)
1380 NEXT I
1390 GOTO 310
1400 REM *****KEEP AS FILE*****
1410 INPUT "CASSETTE/DISK (C/D): ":CD$
1420 IF CD$="D" THEN 1450
1430 OPEN #2:"CS1",OUTPUT,INTERNAL,
     FIXED 128
1440 GOTO 1470
1450 INPUT "OUTPUT FILE NAME: ":F$
1460 OPEN #2:F$
1470 FOR I=1 TO 500
1480 IF I>LAST1 THEN 1520
1490 IF A$(I)="@" THEN 1510
1500 PRINT #2:A$(I)
1510 NEXT I
1520 PRINT #2:"END"
1530 CLOSE #2
1540 GOTO 310
1550 REM *****LIST WORKFILE
     UNNUMBERED*****
1560 FOR I=1 TO 500
1570 IF I>LAST1 THEN 1610
1580 IF A$(I)="@" THEN 1600
1590 PRINT A$(I)
1600 NEXT I
1610 GOTO 310
1620 REM *****RENUMBER WORKFILE*****
1630 INPUT "STEP#: ":STEP1
1640 J=0
```

number (1 for a new file) and the desired increment. An increment of 1 means the records added will be numbered 1, 2, 3, etc., whereas an increment of 5 causes numbering of 1, 6, 11, etc.

When adding new records, you can insert them where desired by renumbering the file and by leaving empty record number gaps. You can then request to add record number 2 after renumbering by a step of 5.

To stop entering records, just key **END**, to terminate the entry and return you to the menu screen (which is just like the EDIT2 menu). To save the data you have entered, key **K** to keep the file. If saving to disk, you will be prompted for the name of the file whose data you want to save. (You might select a name like DSK1.GL83.) If you are writing the data to tape, the screen will prompt you to rewind the tape, press record, etc.

You can also print out the data you have keyed by using the **P** or **PU** command. Then you can see what you have keyed, and you can correct any errors or make additions to the file.

To delete a detail record, you need only key **D** to the menu prompts, followed by the record number range that you want to delete. If you want to change a record, you first need to delete it and then add it back by using the **A** command mentioned earlier.

```
1650 IF STEP1<>0 THEN 1670
1660 STEP1=5
1670 FOR I=1 TO 500
1680 A2$(I)="@"
1690 NEXT I
1700 FOR I=1 TO 500
1710 IF I>LAST1 THEN 1760
1720 IF A$(I)="@" THEN 1750
1730 J=J+STEP1
1740 A2$(J)=A$(I)
1750 NEXT I
1760 FOR I=1 TO 500
1770 A$(I)=A2$(I)
1780 IF A$(I)="@" THEN 1800
1790 LAST1=I
1800 NEXT I
1810 GOTO 310
1820 REM *****PRINT WORKFILE*****
1830 OPEN #3:"RS232.BA=1200"
1840 FOR I=1 TO 500
1850 IF I>LAST1 THEN 1900
1860 IF A$(I)="@" THEN 1890
1870 REM ***VARIOUS LINES FOR LENGTH
     OF A***
1880 PRINT #3:I;TAB(5);A$(I)
1890 NEXT I
1900 CLOSE #3
1910 GOTO 310
1920 REM *****PRINT WORKFILE
     UNNUMBERED*****
1930 OPEN #3:"RS232.BA=1200"
1940 FOR I=1 TO 500
1950 IF I>LAST1 THEN 1990
1960 IF A$(I)="@" THEN 1980
1970 PRINT #3:A$(I)
1980 NEXT I
1990 CLOSE #3
2000 GOTO 310
2010 END
```

GENLED2

```
100 REM * ************************
110 REM *                        *
120 REM *        GENLED2         *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 REM DETAIL$ LAYOUT:
180 REM AC#  DATE  PYE SRC AMT
190 REM ###XX/XX/XX###XXXX#####...
200 SIZEA=100
210 DIM ACCT$(100)
220 DIM PAY$(100)
230 DIM PAYM(100)
240 DIM PAYY(100)
250 REM BELOW 3 DIGIT ACCT #
260 REM FOLLOWED BY + OR -
270 REM FOLLOWED BY ACCT DESC
280 FOR I=1 TO SIZEA
290 PAYM(I)=0
300 PAYY(I)=0
310 NEXT I
320 REM ********************
330 DATA "001-ACCOUNTING, LEGAL"
340 DATA "002-ADVERTISING"
350 DATA "005-AUTO EXPENSE"
360 DATA "010-BAD DEBTS"
370 DATA "012-DEPRECIATION"
380 DATA "015-DONATIONS"
390 DATA "020-EMPLOYEE RETIREMENT"
400 DATA "021-EMPLOYEE BENEFITS"
410 DATA "022-EMPLOYEE SALARIES"
420 DATA "025-INSURANCE"
430 DATA "030-INTEREST PAID"
440 DATA "035-LICENSE"
450 DATA "040-OFFICE/POSTAGE"
460 DATA "042-RENT"
```

GENLED2 reads the data keyed from GENLED1 and produces reports from that data. As discussed in the previous program, GENLED1 keys the detail records and updates already existing files as new data becomes available. Be sure to key the data in the proper order. For instance, you will want to maintain your input in account number by date order. If, by chance, your file size gets too large for GENLED1 to handle easily, you can summarize the year-to-date activity for each account number/payee with one transaction in a new file, preceding the new details.

Now let's go through the following example.

1. To start a new file, run the GENLED1 program.
2. Type **A** after the **/** to add new records when the menu screen appears (described below).

```
        *GENLED1*

A       ADD LINES
C       CLEAR WORKFILE
D       DELETE LINES
T       TEXT IN A FILE
L       LIST WORKFILE
K       KEEP A FILE
LU      LIST UNNUMBERED
R       RENUMBER WORKFILE
P       PRINT WORKFILE
PU      PRINT UNNUMBERED
E       END PROGRAM
```

```
470 DATA "044-REPAIRS/MAINTENANCE"
480 DATA "046-SELLING EXPENSE"
490 DATA "047-TELEPHONE"
500 DATA "048-TRAVEL/ENTERTAINMENT"
510 DATA "049-WARRANTIES"
520 DATA "070-FED INCOME CORP"
530 DATA "071-FED FICA CORP"
540 DATA "073-FED UNEMPLOYMENT"
550 DATA "080-INDIANA INCOME CORP"
560 DATA "082-INDIANA UNEMPLOYMENT"
570 DATA "083-INDIANA SALES TAX"
580 DATA "084-INDIANA REAL PROPERTY"
590 DATA "085-INDIANA MISC"
600 DATA "087-KY MISC"
610 DATA "088-KY SALES TAX"
612 DATA "100+INCOME"
620 DATA "999-NO DESCRIPTION"
630 REM ********************
640 REM FOLLOWING ARE PAYEES
650 DATA "001-PAYEE # 1"
660 DATA "002-PAYEE # 2"
670 DATA "003-PAYEE # 3"
680 DATA "999-NO DESCRIPTION"
690 REM ********************
700 REM ACCT$=ACCT # INFO
710 REM PAY$=PAYEE INFO
720 REM DETAIL$=DETAIL INFO
730 REM A2$=DETAIL ACCT #
740 REM MY$=DETAIL MO/DY/YR
750 REM PN$=DETAIL PAYEE ###
760 REM AMT$=DETAIL AMT
770 REM ********************
780 FOR I=1 TO SIZEA
790 READ X$
800 ACCT$(I)=X$
810 IF SEG$(X$,1,3)="999" THEN 830
820 NEXT I
830 REM *END OF ACCT# READ*
840 FOR I=1 TO SIZEA
850 READ X$
```

3. Key **1** to the "LINE#:" prompt; key **1** to the "STEP#:" prompt.
4. Start entering data by hitting **ENTER** after each field is keyed.

ACCT# (or END):	001
DATE (MM/DD/YY):	01/01/83
PAYEE#:	001
SOURCE:	101
AMOUNT:	5500

After you key the amount (remember that 5500 means fifty-five dollars, since two decimal positions are implied), you will be prompted for the account number of "REC# 2."

Continue in the same manner until you have entered all the records desired. As you finish each record, the data is stored in the A$ array of GENLED1 in the following format:

```
ACCT DATE PYE SRC AMOUNT
###01/01/83###XXXX.....
```

Notice that each field immediately follows the previous one, with no spaces in between. However, for maximum readability, the data that follows, from records 1 to 17, has one space between each field:

```
001 01/01/83 001 101 5500
001 01/15/83 002 105 4000
001 02/10/83 001 115 5098
001 02/20/83 001 120 5500
001 03/08/83 002 130 4650
001 03/25/83 001 138 5400
002 01/01/83 003 102 2500
002 01/24/83 003 107 2000
002 02/25/83 003 122 2325
002 03/17/83 003 135 2575
```

```
 860 PAY$(I)=X$
 870 IF SEG$(X$,1,3)="999" THEN 890
 880 NEXT I
 890 REM *END OF PAYEE# READ*
 900 REM ********************
 910 INPUT "CASSETTE OR DISK: ":RE$
 920 IF RE$="D" THEN 960
 930 OPEN #2:"CS1",INPUT ,INTERNAL,FIXED 128
 940 GOTO 990
 950 REM ********************
 960 INPUT "DSK1.FILENAME: ":FN$
 970 OPEN #2:FN$,INPUT
 980 REM ********************
 990 CALL CLEAR
1000 ACTION$="A"
1010 INPUT "MM/YY DESIRED: ":MMYY$
1020 YYMM$=SEG$(MMYY$,4,2)&SEG$
     (MMYY$,1,2)
1030 INPUT "LIST DETAILS? (Y/N): ":DYN$
1040 INPUT "PRINTER (Y/N): ":PYN$
1050 IF PYN$<>"Y" THEN 1240
1060 OPEN #1:"RS232.BA=1200"
1070 GOSUB 1090
1080 GOTO 1240
1090 PRINT #1:CHR$(12)
1100 IF ACTION$="A" THEN 1150
1110 PRINT #1:TAB(30);"SUMMARY BY PAYEE"
1120 PRINT #1
1130 PRINT #1:"PAYEE";TAB(20);"SRC";TAB(25);
     "ACCOUNT";TAB(64);MMYY$;TAB(70);"YTD"
1140 GOTO 1180
1150 PRINT #1:TAB(30);"SUMMARY BY
     ACCOUNT #"
1160 PRINT #1
1170 PRINT #1:"ACCOUNT";TAB(20);"SRC";
     TAB(25);"PAYEE";TAB(64);MMYY$;
     TAB(70);"YTD"
1180 FOR I=1 TO 79
1190 PRINT #1:"-";
1200 NEXT I
```

```
002 03/19/83 003 136 2480
002 03/28/83 003 140 2900
005 01/01/83 999 103 5500
005 01/11/83 002 104 5350
005 02/05/83 999 113 7500
005 03/09/83 002 131 45040
005 03/25/83 999 139 5000
END
```

After keying **END** for the ACCT# following the last transaction, you key **K** after the **/** to keep the file on cassette or disk. Key **C** or **D** to the "CASSETTE or DISK (C/D):" prompt. If you are keeping to disk, you will be asked for the output file name, to which you key whatever name you choose, for example, **DSK1.GL83**. To keep the file to tape, key **C** and respond to the prompts of "REWIND CASETTE...," "PRESS RECORD...," etc.

When you use GENLED1, you can also list the file you have stored in memory by keying either **LU** to list unnumbered records or **L** to reveal the record numbers. The commands **PU** and **P** will do the same thing to the printer.

At this time you could print your report using GENLED2, but let's say that you left out a record, included one you shouldn't have, and made an error on a third. Either you can make those changes right now, or days later you can text in the same file from tape or disk by using the **T** command of GENLED1.

To text in a file, key **T** and answer **C** or **D** again (for cassette or disk). Then tell the program the disk file name (**DSK1.GL83**, for example) or respond to the prompts of "REWIND...," "PLAY...," etc., for a tape file. You will next be asked for the "STEP#:"—to

```
1210 PRINT #1
1220 LINECTR=4
1230 RETURN
1240 L3TOT=0
1250 DETAILCTR=0
1260 LYMTOT=0
1270 L3GTOT=0
1280 LYMGTOT=0
1290 LAST3$="999"
1300 LASTYYMM$="9999"
1310 FOR I=1 TO 500
1320 INPUT #2:DETAIL$
1325 IF DETAIL$<>"END" THEN 1340
1326 GOSUB 2290
1330 IF DETAIL$="END" THEN 1930
1340 IF SEG$(DETAIL$,1,3)="000" THEN 1920
1350 THISYYMM$=SEG$(DETAIL$,10,2)&SEG$
     (DETAIL$,4,2)
1360 THIS3$=SEG$(DETAIL$,1,3)
1370 LD=LEN(DETAIL$)
1380 LD2=LD-18
1390 AMT1$=SEG$(DETAIL$,19,LD2)
1400 AMT1=VAL(AMT1$)
1410 SRC$=SEG$(DETAIL$,15,4)
1420 IF THIS$="999" THEN 1450
1430 IF LAST3$=THIS3$ THEN 1460
1440 IF LAST3$="999" THEN 1460
1450 GOSUB 2290
1460 LAST3$=THIS3$
1470 LASTYYMM$=THISYYMM$
1480 PAYX$=SEG$(DETAIL$,12,3)
1490 FOR K=1 TO SIZEA
1500 PAY3$=SEG$(PAY$(K),1,3)
1510 IF PAY3$>=PAYX$ THEN 1530
1520 NEXT K
1530 FOR L=1 TO SIZEA
1540 ACCT3$=SEG$(ACCT$(L),1,3)
1550 IF ACCT3$>=SEG$(DETAIL$,1,3)THEN 1570
1560 NEXT L
1570 LP=LEN(PAY$(K))
```

which you might answer **3** or **5** if you want to be able to insert a few records into their proper order. (Let's key **3** here.) This method will renumber the records as they are read in, but the same thing can be done again and again by using the **R** (RE-NUMBER) command.

Now let's add a record before the second record (001 01/15/83 002 105 4000) with the information that follows:

001 01/16/83 001 106 2500

You can first list the resequenced data by keying **L** or **P**, which shows the gaps that have been left. Then you key **A** followed by the "new" line number to be added (**2**) and a step of **1**. You enter the ACCT#, DATE, PAYEE#, SOURCE, and AMOUNT fields, just as before. An **L** command will then show the new data in proper sequence.

To delete record number 4 (originally the third record as resequenced), key **D** to the **/**, followed by **4** for the "BEG LINE#:" and **4** for the "END LINE#:". Notice that when you delete a record, the record after the one deleted is displayed to the screen. Now an **L** command shows that record 4 is gone.

To correct a record—for instance, the last record (now renumbered to 49)—first delete it (**D** and **49** for the beginning and ending numbers) and then add it back correctly using the **A** command. Now re-enter it as

005 03/26/83 999 139 5000

by keying **A**, then **49** for the "LINE#:" and **1** for the "STEP#:" followed by the new data and **END.**

```
1580 LA=LEN(ACCT$(L))
1590 IF SEG$(ACCT$(L),4,1)<>"-" THEN 1610
1600 AMT1=AMT1*(-1)
1610 IF THIS3$="999" THEN 1930
1620 LASTACCT$=ACCT$(L)
1630 LASTPAY$=PAY$(K)
1640 IF THISYYMM$>YYMM$ THEN 1680
1650 L3TOT=L3TOT+AMT1
1660 L3GTOT=L3GTOT+AMT1
1670 PAYY(K)=PAYY(K)+AMT1
1680 IF THISYYMM$<>YYMM$ THEN 1920
1690 LYMTOT=LYMTOT+AMT1
1700 LYMGTOT=LYMGTOT+AMT1
1710 PAYM(K)=PAYM(K)+AMT1
1720 IF DYN$<>"Y" THEN 1920
1730 NUMED=AMT1
1740 MASK$="XXX,XXX.XX-"
1750 GOSUB 2570
1760 IF SEG$(DETAIL$,12,3)=SEG$(PAY$(K),1,3)
     THEN 1790
1770 STR1$=SEG$(DETAIL$,12,3)&"-NO
     DESCRIPTION"
1780 GOTO 1800
1790 STR1$=SEG$(DETAIL$,12,3)&"-"&SEG$
     (PAY$(K),5,LP-4)
1800 IF LINECTR<50 THEN 1820
1810 GOSUB 1090
1820 DETAILCTR=DETAILCTR+1
1830 IF PYN$<>"Y" THEN 1890
1840 IF DETAILCTR>1 THEN 1870
1850 PRINT #1
1860 LINECTR=LINECTR+1
1870 PRINT #1:TAB(10);SEG$(DETAIL$,4,8);TAB(20);
     SEG$(DETAIL$,15,4);TAB(25);STR1$;TAB(50);
     MASK$
1880 LINECTR=LINECTR+1
1890 PRINT SEG$(DETAIL$,4,8);" ";SEG$
     (DETAIL$,15,4)
1900 PRINT STR1$
1910 PRINT MASK$
```

Again, you *must* keep the file to cassette or disk (the same file name unless you purposely don't want to overlay the original file). As with all files, you should keep a backup copy of any file worth saving. Now that you have the proper input data, you are ready to run the report using GENLED2. First, of course, you must end GENLED1 by keying **E** after the */*. (Be sure you have "kept" your file first.)

Running GENLED2 is fairly easy. Once you get familiar with entering and changing files using GENLED1, both programs should be quite simple to use. You may need to step through these detailed instructions to become thoroughly at ease with the file creation. In running GENLED2, you are first asked if the input file is on "CASSETTE or DISK:". As usual, key **C** or **D** and, if on disk, the filename, such as **DSK1.GL83**.

Next are requested the months and year you want on the report ("MM/YY DESIRED:"), to which you key, for example, **03/83**. Then "LIST DETAILS? (Y/N):" asks whether you want to have only ACCT# totals or all the details for the month requested. Answer **Y** for the details or **N** for totals only.

"PRINTER (Y/N):" allows you to print the report either on the screen only or to a printer. Now, as the file is read, the report is displayed either on the screen, or on both the screen and your printer. On the screen each transaction is displayed with the date and source code, followed by the payee number and description, and the amount. The next records are displayed in the same fashion. When the account number

```
1920 NEXT I
1930 L3TOT=L3GTOT
1940 LYMTOT=LYMGTOT
1942 PAYMTOT=0
1944 PAYYTOT=0
1950 CLOSE #2
1960 GOSUB 2480
1970 IF PYN$<>"Y" THEN 2000
1980 PRINT #1
1990 PRINT #1:"*TOTAL*";TAB(59);LYMTOT$;
     TAB(70);L3TOT$
2000 PRINT "*TOT*";LYMTOT$;" ";L3TOT$
2010 IF PYN$<>"Y" THEN 2090
2020 PRINT #1:CHR$(12)
2030 PRINT #1:"PAYEE SUMMARY FOR - ";
     MMYY$;TAB(62);"CURRENT";TAB(77);"YTD"
2040 FOR I=1 TO 79
2050 PRINT #1:"-";
2060 NEXT I
2070 PRINT #1
2080 PRINT #1
2090 FOR I=1 TO SIZEA
2100 IF PAYM(I)<>0 THEN 2120
2110 IF PAYY(I)=0 THEN 2250
2120 NUMED=PAYY(I)
2130 MASK$="XXX,XXX.XX-"
2140 GOSUB 2570
2150 L3TOT$=MASK$
2160 NUMED=PAYM(I)
2170 MASK$="XXX,XXX.XX-"
2180 GOSUB 2570
2190 LYMTOT$=MASK$
2192 PAYMTOT=PAYMTOT+PAYM(I)
2194 PAYYTOT=PAYYTOT+PAYY(I)
2200 IF PYN$<>"Y" THEN 2220
2210 PRINT #1:"PAYEE# ";PAY$(I);TAB(59);
     LYMTOT$;TAB(70);L3TOT$
2220 PRINT "PAYEE# ";PAY$(I)
2230 PRINT MMYY$;"";LYMTOT$
```

(Continued in next column)

changes, a total for the account is given. And at the end of the report, totals are given for the month and year-to-date, as well as totals for each payee number.

One final feature of GENLED2 is important to mention. A minus sign (-) between the account number and the description in the **DATA** statements will cause values to be kept as negative totals for that account, but a plus sign (+) will cause positive totals. This can be shown by adding the following final record to the file above:

```
100 03/30/83 999 SALE 150000
```

Now see the final report on page 132.

(Continued from previous column)

```
2240 PRINT "YTD";L3TOT$
2250 NEXT I
2261 NUMED=PAYMTOT
2262 MASK$="XXX,XXX.XX-"
2263 GOSUB 2570
2264 PAYM$=MASK$
2265 NUMED=PAYYTOT
2266 MASK$="XXX,XXX.XX-"
2267 GOSUB 2570
2268 PAYY$=MASK$
2269 IF PYN$<>"Y" THEN 2275
2270 PRINT #1
2271 PRINT #1:TAB(59);PAYM$;TAB(70);PAYY$
2274 CLOSE #1
2275 PRINT "PAYEE MO TOT:";PAYM$
2276 PRINT "PAYEE YR TOT:";PAYY$
2280 GOTO 2920
2290 REM ********************
```

```
2300 IF LAST3$=SEG$(LASTACCT$,1,3)THEN 2330
2310 ACCTDESC$=LAST3$&"-NO DESCRIPTION"
2320 GOTO 2340
2330 ACCTDESC$=LAST3$&"-"&SEG$
     (LASTACCT$,5,LA-4)
2340 GOSUB 2480
2350 IF PYN$<>"Y" THEN 2390
2360 IF LINECTR<50 THEN 2380
2370 GOSUB 1090
2380 PRINT #1:ACCTDESC$;TAB(59);LYMTOT$;
     TAB(70);L3TOT$
2390 PRINT ACCTDESC$
2400 PRINT LYMTOT$;" ";L3TOT$
2410 PRINT "**************"
2420 LINECTR=LINECTR+1
2430 L3TOT=0
2440 LYMTOT=0
2450 DETAILCTR=0
2460 PRINT$="Y"
2470 RETURN
2480 NUMED=LYMTOT
2490 MASK$="XXX,XXX.XX-"
2500 GOSUB 2570
2510 LYMTOT$=MASK$
2520 NUMED=L3TOT
2530 MASK$="XXX,XXX.XX-"
2540 GOSUB 2570
2550 L3TOT$=MASK$
2560 RETURN
2570 REM ***EDIT1***
2580 REM ***EDITING SUBROUTINE***
2590 REM ***MASK$=MASK FOR EDITED
     RESULT
2600 REM ***EXAMPLE: XXX,XXX.XX-
2610 REM ***NUMED=NUMBER TO EDIT
     (INTEGER FORM)
2620 IF NUMED<>0 THEN 2650
2630 MASK$=" "
2640 GOTO 2910
```

(Continued in next column)

(Continued from previous column)

```
2650 NUMED2=NUMED
2660 NUMED=ABS(NUMED)
2670 LMASK=LEN(MASK$)
2680 NUMED$=STR$(NUMED)
2690 LNUMED=LEN(NUMED$)
2700 CTN=LNUMED+1
2710 FOR CTM=LMASK TO 1 STEP -1
2720 CTN=CTN-1
2730 IF CTN<1 THEN 2890
2740 IF SEG$(MASK$,CTM,1)<>"-" THEN 2820
2750 IF NUMED2>0 THEN 2790
2760 MASK$=SEG$(MASK$,1,CTM-1)&"-"
2770 CTN=CTN+1
2780 GOTO 2810
2790 MASK$=SEG$(MASK$,1,CTM-1)&" "
2800 CTN=CTN+1
2810 GOTO 2900
2820 IF SEG$(MASK$,CTM,1)="," THEN 2850
2830 IF SEG$(MASK$,CTM,1)="." THEN 2850
2840 GOTO 2870
2850 CTN=CTN+1
2860 GOTO 2900
2870 MASK$=SEG$(MASK$,1,CTM-1)&SEG$
     (NUMED$,CTN,1)&SEG$(MASK$,CTM+1,
     (LMASK-(CTM-1)-1))
2880 GOTO 2900
2890 MASK$=SEG$(MASK$,1,CTM-1)&" "&SEG$
     (MASK$,CTM+1,(LMASK-(CTM-1)-1))
2900 NEXT CTM
2910 RETURN
2920 END
```

SUMMARY BY ACCOUNT #

ACCOUNT	SRC	PAYEE		03/83	YTD
03/08/83	130	002-PAYEE # 2	46.50-		
03/25/83	138	001-PAYEE # 1	54.00-		
001-ACCOUNTING, LEGAL				100.50-	286.48-
03/17/83	135	003-PAYEE # 3	25.75-		
03/19/83	136	003-PAYEE # 3	24.80-		
03/28/83	140	003-PAYEE # 3	29.00-		
002-ADVERTISING				79.55-	147.80-
03/09/83	131	002-PAYEE # 2	450.40-		
03/26/83	139	999-NO DESCRIPTION	50.00-		
005-AUTO EXPENSE				500.40-	683.90-
03/30/83	SALE	999-NO DESCRIPTION	1,500.00		
100-INCOME				1,500.00	1,500.00
TOTAL				819.55	381.82

PAYEE SUMMARY FOR - 03/83	CURRENT	YTD
PAYEE# 001-PAYEE # 1	54.00-	239.98-
PAYEE# 002-PAYEE # 2	496.90-	550.40-
PAYEE# 003-PAYEE # 3	79.55-	147.80-
PAYEE# 999-NO DESCRIPTION	1,450.00	1,320.00
	819.55	381.82

```
00101/01/83001101 5500
00101/16/83001106 2500
00102/10/83001115 5098
00102/20/83001120 5500
00103/08/83002130 4650
00103/25/83001138 5400
00201/01/83003102 2500
00201/24/83003107 2000
00202/25/83003122 2325
00203/17/83003135 2575
00203/19/83003136 2480
00203/28/83003140 2900
00501/01/83999103 5500
00501/01/83002104 5350
00502/05/83999113 7500
00503/09/83002131 45040
00503/26/83999139 5000
10003/30/83999SALE150000
```

GRADES

```
100 REM * ************************
110 REM *                        *
120 REM *        GRADES          *
130 REM *                        *
140 REM *     COPYRIGHT 1983     *
150 REM *   DONALD C. KREUTNER   *
160 REM * ************************
170 OPTION BASE 1
180 DIM N$(40)
190 DIM S$(40,20)
200 DIM N(40)
210 DIM T(40)
220 REM ************************
230 CALL CLEAR
240 PRINT "CLEARING MEMORY"
250 FOR I=1 TO 40
260 N(I)=I
270 T(I)=0
280 FOR J=1 TO 20
290 S$(I,J)="@"
300 NEXT J
310 NEXT I
320 REM ************************
330 CALL CLEAR
340 PRINT "NAME/SCORE FILE INPUT"
350 INPUT "CASSETTE OR DISK (C/D): ":CD$
360 IF CD$="C" THEN 410
370 IF CD$<>"D" THEN 330
380 INPUT "DSK1.FILENAME: ":D$
390 OPEN #1:D$
400 GOTO 420
410 OPEN #1:"CS1",INTERNAL,FIXED 128,INPUT
420 CTRN=0
430 CTRD=0
440 INPUT #1:X$
450 IF X$="END" THEN 600
460 X1$=SEG$(X$,1,1)
```

GRADES is a program that will keep total grade scores for a class of up to 40 students. The file that keeps the data can be kept on tape or disk and altered as new data becomes available.

The input file is created using EDIT1 or EDIT2. Sample data is as follows:

```
*PERFECT SCORE
100
100
150
150
DOE JOHN
99
97
134
145
DOE JANE
100
98
125
149
END
```

After data has been entered, GRADES can then process it. Each name record is followed by the scores for test (or quiz) number 1, 2, 3, etc. The first record "*PERFECT SCORE" is recognized as the master score record because of the asterisk in the first position. Any record that starts with an asterisk (*) or an alphabetical letter causes a new accumulation of the records that follow that name record (lines 470-500). Any other record that does not begin with a number is rejected (lines 510-520), and the next record is read.

```
470 IF X1$<"A" THEN 500
480 IF X1$>"Z" THEN 440
490 GOTO 560
500 IF X1$="*" THEN 560
510 IF X1$<"0" THEN 440
520 IF X1$>"9" THEN 440
530 CTRD=CTRD+1
540 S$(CTRN,CTRD)=X$
550 GOTO 440
560 CTRN=CTRN+1
570 CTRD=0
580 N$(CTRN)=X$
590 GOTO 440
600 CALL CLEAR
610 CLOSE #1
620 PRINT "SORTING BY NAME..."
630 PASS=0
640 FL$="N"
650 PASS=PASS+1
660 PRINT "SORT PASS";PASS
670 FOR I=1 TO CTRN-1
680 IF N$(I)<=N$(I+1)THEN 760
690 NX$=N$(I)
700 N$(I)=N$(I+1)
710 N$(I+1)=NX$
720 NX=N(I)
730 N(I)=N(I+1)
740 N(I+1)=NX
750 FL$="Y"
760 NEXT I
770 IF FL$="Y" THEN 640
780 PRINT "SORT COMPLETE..."
790 CALL CLEAR
800 INPUT "PAUSE (Y/N): ":PAYN$
810 INPUT "PRINTER (Y/N): ":PYN$
820 IF PYN$<>"Y" THEN 870
830 OPEN #2:"RS232.BA=1200"
840 INPUT "TITLE: ":TI$
850 PRINT #2:TAB(5);TI$
```

(Continued in next column)

As in other programs, the input file can be from tape or disk (lines 350-430). After all the data is read in and the END record is encountered, a sort of the name records takes place. The score details are kept in the array S$, whereas the array N keeps track of that part of the array where the scores for the sorted names are located after the sort (lines 620-780).

When the sort is ended, you are then asked if you want a pause, which causes the display to delay until you hit **ENTER** after each student's scores. You can also direct the report to a printer or to the screen only (lines 800-860).

The scores are printed left to right to save space on the screen and on the printer. And in order to change or add to the data, you can modify the input file by using one of the edit programs.

One last alternative is to save the sorted data to a new file. Then you can add new students at the end of your data and save the sorted file without having to insert many new records in the middle of the file.

(Continued from previous column)

```
860 PRINT #2
870 FOR I=1 TO CTRN
880 TOTI=0
890 I$=STR$(I)
900 PRINT I$;".";TAB(5);N$(I)
910 IF PYN$<>"Y" THEN 930
920 PRINT #2:I$;".";TAB(5);N$(I)
```

```
 930 FOR J=1 TO 20
 940 IF S$(N(I),J)="@" THEN 1010
 950 PRINT S$(N(I),J);" ";
 960 IF PYN$<>"Y" THEN 980
 970 PRINT #2:S$(N(I),J);" ";
 980 SCORE1=VAL(S$(N(I),J))
 990 TOTI=TOTI+SCORE1
1000 NEXT J
1010 PRINT
1020 T(I)=TOTI
1030 PRINT "*TOTAL*";TOTI
1040 PRINT
1050 IF PYN$<>"Y" THEN 1090
1060 PRINT #2
1070 PRINT #2:"*TOTAL*";TOTI
1080 PRINT #2
1090 IF PAYN$<>"Y" THEN 1110
1100 INPUT "ENTER:":CO$
1110 NEXT I
1120 IF PYN$<>"Y" THEN 1140
1130 CLOSE #2
1140 INPUT "REPEAT (Y/N): ":YN$
1150 IF YN$="Y" THEN 790
1160 CALL CLEAR
1170 INPUT "SAVE NEW FILE (Y/N): ":FYN$
1180 IF FYN$<>"Y" THEN 1340
1190 INPUT "CASSETTE OR DISK (C/D): ":CD$
1200 IF CD$="C" THEN 1250
1210 IF CD$<>"D" THEN 1160
1220 INPUT "DSK1.FILENAME: ":F$
1230 OPEN #1:F$
1240 GOTO 1260
1250 OPEN #1:"CS1",OUTPUT,INTERNAL,
     FIXED 128
1260 FOR I=1 TO CTRN
1270 PRINT #1:N$(I)
1280 PRINT I;N$(I)
1290 PRINT #1:T(I)
1300 PRINT T(I)
```

(Continued in next column)

(Continued from previous column)

```
1310 NEXT I
1320 PRINT #1:"END"
1330 CLOSE #1
1340 END
```

```
*PERFECT SCORE
100
150
100
50
200
DOE JANE
90
130
99
45
187
DOE JOHN
89
135
95
47
180
LARSON JIM
88
140
100
50
190
MILLER CHARLES
90
143
96
46
185
NEILS ROBERT
87
146
97
45
186
PATTERSON KEVIN
80
145
95
40
180
PAYTON RUSSELL
90
140
99
44
184
```

```
    TOTAL SCORES (09/83)

1.  *PERFECT SCORE
100 150 100 50 200
*TOTAL* 600

2.  DOE JANE
90 130 99 45 187
*TOTAL* 551

3.  DOE JOHN
89 135 95 47 180
*TOTAL* 546

4.  LARSON JIM
88 140 100 50 190
*TOTAL* 568

5.  MILLER CHARLES
90 143 96 46 185
*TOTAL* 560

6.  NEILS ROBERT
87 146 97 45 186
*TOTAL* 561

7.  PATTERSON KEVIN
80 145 95 40 180
*TOTAL* 540

8.  PAYTON RUSSELL
90 140 99 44 184
*TOTAL* 557
```

MEMO

```
100 REM * ************************
110 REM *                        *
120 REM *          MEMO          *
130 REM *                        *
140 REM *      COPYRIGHT 1983    *
150 REM *    DONALD C. KREUTNER  *
160 REM *                        *
170 REM * ************************
180 OPTION BASE 1
190 DIM A$(300)
200 REM INPUT DATA FOR MEMO IS KEYED
210 REM BY EDIT1 OR EDIT2 PROGRAM
220 REM RECORD LAYOUT:
230 REM MM/DD/YY HH:MM AM (ANY MEMO
    INFO)
240 CALL CLEAR
250 PRINT "          ***MEMO***"
260 PRINT
270 INPUT "CASSETTE OR DISK: ":CD$
280 IF CD$="C" THEN 330
290 IF CD$<>"D" THEN 240
300 INPUT "DSK1.FILENAME: ":D$
310 OPEN #1:D$
320 GOTO 340
330 OPEN #1:"CS1",INTERNAL,FIXED 128,INPUT
340 FOR I=1 TO 300
350 INPUT #1:A$(I)
360 IF A$(I)="END" THEN 380
370 NEXT I
380 CLOSE #1
390 PASS=0
400 CTRN=I-1
410 CALL CLEAR
420 SW$="N"
430 PASS=PASS+1
440 PRINT "SORT PASS";PASS
450 FOR I=1 TO CTRN-1
```

MEMO is designed to serve as a personal calendar or note pad to remind you of important events and appointments.

The input file is keyed by EDIT1 or EDIT2, programs discussed in Chapter 2. The records you key require the following format:

Pos. 1-8	Date of the memo, by month, day, and year (MM/DD/YY) Example: 01/01/83
Pos. 10-17	Time of the appointment or event (hour:minute followed by AM, PM, or blank) Example: 10:30 AM
Pos. 19 to the end	The appointment, person, or event to be remembered

When you key in these memos, you will most likely add them in chronological order. Remember that if you want to add new records later by using EDIT2, you can insert, delete, and alter records, thus keeping them in proper sequence. You can have many different "memo" files that keep track of different categories. The input data that follows is for the report example below:

```
09/10/83 10:30 AM MEETING AT L.R.D.
09/10/83 12:30 AM LUNCH WITH DAVE
09/10/83  2:30 PM REPORT DUE
09/11/83  9:30 AM CHECK FILES
09/11/83 11:00 AM MEET WITH NEW
CLIENT
```

```
460 S1$=SEG$(A$(I),7,2)&SEG$(A$(I),1,6)&SEG$
    (A$(I),16,1)&SEG$(A$(I),10,10)
470 S2$=SEG$(A$(I+1),7,2)&SEG$(A$(I+1),1,6)
    &SEG$(A$(I+1),16,1)&SEG$(A$(I+1),10,10)
480 IF S1$<=S2$ THEN 530
490 AX$=A$(I)
500 A$(I)=A$(I+1)
510 A$(I+1)=AX$
520 SW$="Y"
530 NEXT I
540 IF SW$="Y" THEN 420
550 PRINT "SORT COMPLETE"
560 CALL CLEAR
570 INPUT "PRINTER (Y/N): ":PYN$
580 INPUT "PAUSE (Y/N): ":PAUS$
590 IF PYN$<>"Y" THEN 630
600 OPEN #2:"RS232.BA=1200"
610 PRINT #2:"  DATE";TAB(13);"TIME";TAB(24);
    "MEMORANDUM"
620 PRINT #2
630 FOR I=1 TO CTRN
640 DT$=SEG$(A$(I),1,8)
650 TM$=SEG$(A$(I),10,8)
660 ME$=SEG$(A$(I),19,LEN(A$(I))-18)
670 PRINT DT$;"  ";TM$
680 PRINT ME$
690 PRINT "**************************"
700 IF PAUS$<>"Y" THEN 720
710 INPUT " ":X$
720 IF PYN$<>"Y" THEN 740
730 PRINT #2:DT$;TAB(12);TM$;TAB(24);ME$
740 NEXT I
750 IF PYN$<>"Y" THEN 770
760 CLOSE #2
770 PRINT
780 INPUT "REPEAT (Y/N): ":RYN$
790 IF RYN$="Y" THEN 560
800 INPUT "REWRITE SORTED FILE
    (Y/N): ":WYN$
810 IF WYN$<>"Y" THEN 950
```

```
09/11/83 1:30 PM BOARD MEETING
09/11/83 7:00 PM ENTERTAIN GUESTS
```

Notice that in keying the time, you must skip two spaces after the date ending in position 8 (or key a zero) in front of a single-hour digit (1:30 PM). So that events are sorted correctly, the AM or PM designation is necessary. Obviously, these fields must be in exactly the same locations, or the sort in lines 410-550 is completely ineffective.

The file that you read in from cassette or disk in lines 270-380 can be in order, in fairly good order, or even in very bad order. The program will sort the memos by date and time if you have keyed in the times in the correct positions. But how long the sort will take, of course, depends on how disordered the data is and on how many memos you have read in. The data is placed in the array A$ (lines 190 and 350).

You are then asked in lines 570 and 580 whether you want a printout or whether you want the program to pause so that you can key **ENTER**, preventing the memos from rolling off the screen.

After the report is completed, you can repeat the report, or before the program ends, you can rewrite the data in sorted order to a new data file, which you can then manipulate with the edit programs.

```
820 INPUT "CASSETTE OR DISK (C/D): ":CD$
830 IF CD$="C" THEN 880
840 IF CD$<>"D" THEN 820
850 INPUT "DSK1.FILENAME: ":D$
860 OPEN #3:D$
870 GOTO 890
880 OPEN #3:"CS1",INTERNAL,FIXED 128,
    OUTPUT
890 FOR I=1 TO CTRN
900 PRINT #3:A$(I)
910 PRINT I
920 NEXT I
930 PRINT #3:"END"
940 CLOSE #3
950 END
```

```
  DATE       TIME        MEMORANDUM

09/10/83   10:30 AM    MEETING AT L.R.D.
09/10/83   12:30 PM    LUNCH WITH DAVE
09/10/83    2:30 PM    REPORT DUE
09/11/83    9:30 AM    CHECK FILES
09/11/83   11:00 AM    MEET WITH NEW CLIENT
09/11/83    1:30 PM    BOARD MEETING
09/11/83    7:00 PM    ENTERTAIN GUESTS
```

PHONEADD

```
100 REM ************************
110 REM *                      *
120 REM *      PHONEADD        *
130 REM *                      *
140 REM *    COPYRIGHT 1983    *
150 REM *  DONALD C. KREUTNER  *
160 REM ************************
170 DATA 812-999-1234 BIRK WILLIAM/3299
    SECOND STREET/KANSAS CITY
180 DATA 812-111-4321 BROWN RALPH
190 DATA 812-222-1111 CRAWFORD MAURICE
195 DATA 812-777-1456 DAUDT JACK
200 DATA 812-123-2134 GERDES JESSE/4960
    WESTMINSTER/LONDON ENGLAND
210 DATA 812-333-2233 HATFIELD DAVID
220 DATA 812-233-2266 HUBLAR AL
230 DATA 987-127-7654 JENSEN MARY
240 DATA 900-435-2222 KOCH WILLIAM
250 DATA 807-213-5555 LANG NEIL
260 DATA 812-333-4441 MANOR JANE
265 DATA 812-222-9963 MARRA ANNA
270 DATA 900-111-3544 MCALLISTER SCOTT
280 DATA 888-222-2133 MEADOR STEVE
290 DATA 344-123-2186 MILLER RONALD
300 DATA 412-210-2211 MILLSPAUGH RICHARD
310 DATA 322-133-4455 SCOTT DAVID
320 DATA 277-900-0001 STEWART CHARLES
330 DATA 244-122-4321 STEWART GARY
340 DATA 345-222-8888 STOKES ROBERT
350 DATA 367-323-4466 TRNKA JOHN
360 DATA 999-233-7777 VAN BRUSSEL
    CORNELIA
370 DATA 888-223-2211 WALTERS DARYLL
380 DATA 723-299-8867 YATES WILLIAM
390 DATA END
400 OPTION BASE 1
410 DIM PH$(200)
```

PHONEADD is a phone/name/address directory program. It is capable of looking up (by last name) an individual's phone number and address, if desired.

In this program the input data can come from three sources (lines 470-490): cassette, disk, or program data (C/D/P). You can, therefore, use many different files to obtain telephone numbers and addresses of various categories of people. And you can put the most frequently used list into the program's **DATA** statements. (See lines 170-390.) Furthermore, if you arrange these **DATA** statements in alphabetical order (as well as the files you create when using EDIT1 or EDIT2), you can obtain a very rapid data "load," since the alphabetical sort in lines 900-1010 will be required to make only one pass through the data to verify that it is in order.

Line 460 allows you to by-pass the phone number sort, again saving valuable time in preparing the program's data for use. This by-pass is very reasonable since in most cases you will want to look up by last name rather than telephone number. If, however, you also want to search by telephone number, the sort in lines 700-850 will accomplish just that.

The actual lookup takes place in lines 1030-1520. If you select the phone number key also, the number or name you key in line 1060 is analyzed so that the program "knows" whether you are looking up by number or name. But if you decline the

```
420 DIM NM$(200)
430 CALL CLEAR
440 PRINT "          ***PHONEADD***"
450 PRINT
460 INPUT "ALPHABETIC ONLY (Y/N): ":AYN$
470 PRINT "CASSETTE OR DISK "
480 PRINT "OR PROGRAM DATA (C/D/P): ";
490 INPUT " ":CD$
500 IF CD$="P" THEN 570
510 IF CD$="C" THEN 560
520 IF CD$<>"D" THEN 430
530 INPUT "DSK1.FILENAME: ":D$
540 OPEN #1:D$
550 GOTO 570
560 OPEN #1:"CS1",INTERNAL,FIXED 128,INPUT
570 FOR I=1 TO 200
580 IF CD$<>"P" THEN 610
590 READ X$
600 GOTO 620
610 INPUT #1:X$
620 IF X$="END" THEN 670
630 IF AYN$="Y" THEN 650
640 PH$(I)=X$
650 NM$(I)=X$
660 NEXT I
670 CTRN=I-1
680 IF CD$="P" THEN 700
690 CLOSE #1
700 PASS=0
710 IF AYN$="Y" THEN 860
720 SW$="N"
730 PASS=PASS+1
740 PRINT "PHONE# SORT PASS";PASS
750 FOR I=1 TO CTRN-1
760 PH1$=SEG$(PH$(I),1,12)
770 PH2$=SEG$(PH$(I+1),1,12)
780 IF PH1$<=PH2$ THEN 830
790 SW$="Y"
800 PHX$=PH$(I)
810 PH$(I)=PH$(I+1)
```

phone number search in line 460, you may look up by name only (line 1040, instead of line 1060).

PHONEADD will display one entry after another as long as you keep hitting **ENTER**, but you may look up any other name or number by keying that name or number.

As mentioned earlier, an entry may or may not contain the address associated with the name. If you want to have the addresses on all or some entries, as in this program, simply key the addresses, where desired, following the name (perhaps separated by slashes).

Records you key in a file are similar to the **DATA** statements in the program:

Pos. 1-12	The telephone number preceded by area code and separated by hyphens (999-999-9999)
Pos. 14 to the end of record	The name and, if desired, an abbreviated address (as short as possible for memory conservation)

Following is an example of an entry for PHONEADD:

```
999-999-9999 WHITE
MAX/9999 WHITESIDE
DRIVE/ANYTOWN USA
```

(All data appears on one line.)

```
 820 PH$(I+1)=PHX$
 830 NEXT I
 840 IF SW$="Y" THEN 720
 850 PRINT "PHONE# SORT COMPLETE"
 860 PASS=0
 870 SW$="N"
 880 PASS=PASS+1
 890 REM ********************
 900 PRINT "NAME SORT PASS";PASS
 910 FOR I=1 TO CTRN-1
 920 NM1$=SEG$(NM$(I),14,LEN(NM$(I))-13)
 930 NM2$=SEG$(NM$(I+1),14,LEN(NM$(I+1))-13)
 940 IF NM1$<=NM2$ THEN 990
 950 SW$="Y"
 960 NMX$=NM$(I)
 970 NM$(I)=NM$(I+1)
 980 NM$(I+1)=NMX$
 990 NEXT I
1000 IF SW$="Y" THEN 870
1010 PRINT "NAME SORT COMPLETE"
1020 CALL CLEAR
1030 IF AYN$<>"Y" THEN 1060
1040 PRINT "PARTIAL/COMPLETE NAME"
1050 GOTO 1070
1060 PRINT "PARTIAL/COMPLETE #/NAME"
1070 PRINT "(OR END TO END PROGRAM)"
1080 INPUT " ":K$
1090 IF K$="END" THEN 1540
1100 LENK=LEN(K$)
1110 IF AYN$<>"Y" THEN 1150
1120 IF SEG$(K$,1,1)<"A" THEN 1020
1130 IF SEG$(K$,1,1)>"Z" THEN 1020
1140 GOTO 1360
1150 IF SEG$(K$,1,1)>="A" THEN 1360
1160 IF SEG$(K$,1,1)=" " THEN 1190
1170 IF SEG$(K$,1,1)<"0" THEN 1020
1180 IF SEG$(K$,1,1)>"9" THEN 1020
1190 FOR I=1 TO CTRN
1200 LENP=LEN(PH$(I))
```

(Continued in next column)

(Continued from previous column)

```
1210 IF LENP>LENK THEN 1230
1220 LENK=LENP
1230 IF SEG$(PH$(I),1,LENK)>=K$ THEN 1280
1240 NEXT I
1250 PRINT "NO RECORD >= ";K$
1260 INPUT " ":X$
1270 GOTO 1020
1280 FOR J=I TO CTRN
1290 PRINT SEG$(PH$(J),1,13)
1300 PRINT SEG$(PH$(J),14,LEN(PH$(J))-13)
1310 PRINT "**************************"
1320 INPUT " ":X$
1330 IF X$="END" THEN 1020
1340 NEXT J
1350 GOTO 1020
1360 IF SEG$(K$,1,1)>"Z" THEN 1020
1370 FOR I=1 TO CTRN
1380 LENN=LEN(NM$(I))
1390 IF LENN>LENK THEN 1410
1400 LENK=LENN
1410 IF SEG$(NM$(I),14,LENK)>=K$ THEN 1460
1420 NEXT I
1430 PRINT "NO RECORD >= ";K$
1440 INPUT " ":X$
1450 GOTO 1020
1460 FOR J=I TO CTRN
1470 PRINT SEG$(NM$(J),1,13)
1480 PRINT SEG$(NM$(J),14,LEN(NM$(J))-13)
1490 PRINT "**************************"
1500 INPUT " ":X$
1510 IF X$="END" THEN 1020
1520 NEXT J
1530 GOTO 1020
1540 END
```

PRICELIST

```
100 REM ************************
110 REM *                      *
120 REM *      PRICELIST       *
130 REM *                      *
140 REM *    COPYRIGHT 1983    *
150 REM *  DONALD C. KREUTNER  *
160 REM ************************
170 REM THIS PROGRAM USES AN EDIT
180 REM FILE THAT IS READ IN OR
190 REM DATA STATEMENTS CAN BE
200 REM USED WITHIN THE PROGRAM.
201 REM PRICE IS IN POS. 1-8
202 REM COST IS IN 10-17
203 REM DESCRIPTION IS IN 19-END
204 REM THE DESCRIPTION'S 1ST
205 REM CHARACTERS ARE THE ITEM #
210 DATA 00005000 00003000 0101 ITEM X
215 DATA 00010000 00005000 0150 ITEM Y
220 DATA 00020000 00010000 0250 ITEM Z
225 DATA 00030000 00015000 0350 ITEM XYZ
230 DATA END
260 OPTION BASE 1
270 DIM P$(250)
280 DIM ITEM$(50)
290 DIM ITEMNO(50)
300 DIM Q(50)
310 P$(1)="END"
320 ITEM$(1)="END"
330 CALL CLEAR
340 PRINT "R    READ PRICE FILE"
350 PRINT "D    READ PROGRAM DATA"
360 PRINT "W    WRITE PRICE FILE"
370 PRINT "S    SELECT PRICE ITEMS"
380 PRINT "L    LIST PRICE ITEMS"
390 PRINT "E    END PROGRAM"
400 PRINT
410 INPUT "SELECTION: ":S$
```

PRICELIST is a versatile program that can provide you with an estimate of prices for a client or costs for the production of an item, based on costs and prices for components or parts that you have given in a file. Or the program can simply give to a customer the total price for a group of items.

The input for the master price list can be program **DATA** statements, or a tape or disk file created by EDIT1 or EDIT2. (See the menu screen in lines 340-400.) The file record layout is as follows:

Pos. 1-8	The price (needs leading zeros) Example: $45.75 is 00004575
Pos. 10-17	The cost (needs leading zeros)
Pos. 19 to end of record	Item number and description

PRICELIST does not require you to have a certain kind of item number (specified as so many characters long, composed of leading alphabetical characters, etc.). Instead, the item number can simply be the leading characters of the description. Then, when you later key an item number to be priced, the number keyed is searched for in the ITEM$ array, by the length of the number you enter. If there is no match for the item number keyed, you are asked to rekey it for a correct match (at the time you key the item number, not when you have finished and requested a report).

```
420 IF S$="R" THEN 500
430 IF S$="D" THEN 590
440 IF S$="W" THEN 880
450 IF S$="S" THEN 1010
460 IF S$="L" THEN 1210
470 IF S$="E" THEN 2240
480 GOTO 410
490 REM *************************
500 INPUT "CASSETTE OR DISK: ":CD$
510 IF CD$="C" THEN 560
520 IF CD$<>"D" THEN 500
530 INPUT "DSK1.FILENAME: ":D$
540 OPEN #1:D$
550 GOTO 600
560 OPEN #1:"CS1",INTERNAL,FIXED 128,INPUT
570 IF S$<>"D" THEN 600
580 REM *************************
590 RESTORE
600 FOR I=1 TO 250
610 IF S$<>"D" THEN 640
620 READ P$(I)
630 GOTO 650
640 INPUT #1:P$(I)
650 IF P$(I)="END" THEN 670
660 NEXT I
670 CTRP=I-1
680 IF S$="D" THEN 700
690 CLOSE #1
700 IF CTRP>0 THEN 720
710 CTRP=1
720 PASS=0
730 FL$="N"
740 PASS=PASS+1
750 CALL CLEAR
760 PRINT "SORT PASS ";PASS
770 FOR I=1 TO CTRP-1
780 IF SEG$(P$(I),19,LEN(P$(I))-18)<=SEG$(P$(I+1),
    19,LEN(P$(I+1))-18)THEN 830
790 FL$="Y"
800 P2$=P$(I)
```

In PRICELIST you simply key each item number and quantity in lines 1020-1170 until you are finished. Then you key **END** for the item number and return to the menu screen. At that time you can request a printout or screen listing of the prices you have just keyed.

Lines 1210-2010 produce the price list of selected parts. You can select price or cost, or both (line 1220). In line 1310 you can supply a customer name or any other suitable description.

A pause is built in to line 1760 (which you can remove if you want only to print these lists to the printer).

Again, the program will run more smoothly if you input a price file that is sorted by item number, although the program will sort the data (lines 720-860) if the items are out of sequence. Also, if you use the same price list again and again, you should use **DATA** statements to replace those now in the program. Just be sure to end them with an END record. Following is an example of a price list file:

```
00040000 00027500 A001
CEDAR BOX

00000995 00000545 A002
HAMMER

00039795 00022000 B010
COLOR TELEVISION

00001095 00000650 C0098
SHIN GUARDS

END
```

```
 810 P$(I)=P$(I+1)
 820 P$(I+1)=P2$
 830 NEXT I
 840 IF FL$="Y" THEN 730
 850 PRINT "SORT COMPLETE"
 860 GOTO 330
 870 REM **************************
 880 INPUT "CASSETTE OR DISK: ":CD$
 890 IF CD$="C" THEN 940
 900 IF CD$<>"D" THEN 880
 910 INPUT "DSK1.FILENAME: ":D$
 920 OPEN #1:D$
 930 GOTO 950
 940 OPEN #1:"CS1",INTERNAL,FIXED 128,
     OUTPUT
 950 FOR I=1 TO CTRP+1
 960 PRINT #1:P$(I)
 970 NEXT I
 980 CLOSE #1
 990 GOTO 330
1000 REM **************************
1010 CALL CLEAR
1020 FOR I=1 TO 49
1030 PRINT I;TAB(4);"ITEM# (OR END):";
1040 INPUT " ":ITEM$(I)
1050 ITEMNO(I)=0
1060 IF ITEM$(I)="END" THEN 330
1070 LENI=LEN(ITEM$(I))
1080 FOR J=1 TO CTRP
1090 IF SEG$(P$(J),19,LENI)<>ITEM$(I)THEN 1120
1100 ITEMNO(I)=J
1110 GOTO 1130
1120 NEXT J
1130 IF ITEMNO(I)>0 THEN 1160
1140 PRINT CHR$(7)
1150 GOTO 1030
1160 INPUT "QTY: ":Q(I)
1170 NEXT I
1180 ITEM$(I)="END"
```

(Continued in next column)

Notice that the item number C0098 is longer than the others. This difference does not really matter. When you key **C0098**, the program will look for an item number of exactly that length. In fact, some item numbers could be one character long, whereas other numbers in the same file might consist of 20 or more characters, although this is unlikely.

You can also write out the sorted price file to a tape or disk file to speed up the sort process at program startup. Of course, if your items are already in ordered **DATA** statements, you will not need to do this step.

(Continued from previous column)

```
1190 GOTO 330
1200 REM **************************
1210 CALL CLEAR
1220 INPUT "PRICE/COST/BOTH (P/C/B): ":PCB$
1230 IF PCB$="P" THEN 1260
1240 IF PCB$="C" THEN 1260
1250 IF PCB$<>"B" THEN 1210
1260 INPUT "PRINTER (Y/N): ":PYN$
1270 TOTC=0
1280 TOTP=0
1290 IF PYN$<>"Y" THEN 1410
1300 OPEN #2:"RS232.BA=1200"
1310 INPUT "CUSTOMER NAME: ":CN$
1320 PRINT #2:CHR$(12)
1330 PRINT #2:CN$
1340 PRINT #2
1350 IF PCB$="C" THEN 1370
1360 PRINT #2:"PRICE";
1370 IF PCB$="P" THEN 1390
```

```
1380 PRINT #2:TAB(12);"          COST";
1390 PRINT #2:TAB(24);"QTY";TAB(30);
     "ITEM#/DESCRIPTION"
1400 PRINT #2
1410 CTRS=0
1420 FOR I=1 TO 50
1430 IF ITEM$(I)="END" THEN 1790
1440 ITEMDESC$=SEG$(P$(ITEMNO(I)),19,
     LEN(P$(ITEMNO(I))))
1450 PRINT I;TAB(5);ITEMDESC$
1460 P1$=SEG$(P$(ITEMNO(I)),1,8)
1470 P1=VAL(P1$)
1480 C1$=SEG$(P$(ITEMNO(I)),10,8)
1490 C1=VAL(C1$)
1500 P1=Q(I)*P1
1510 C1=Q(I)*C1
1520 Q$=STR$(Q(I))
1530 PRINT Q$;TAB(6);
1540 IF PCB$="C" THEN 1590
1550 AMT1=P1
1560 GOSUB 2030
1570 P2$=AMT2$
1580 PRINT P2$;" ";
1590 IF PCB$="P" THEN 1640
1600 AMT1=C1
1610 GOSUB 2030
1620 C2$=AMT2$
1630 PRINT C2$;" *";
1640 PRINT
1650 PRINT
1660 IF PYN$<>"Y" THEN 1720
1670 IF PCB$="C" THEN 1690
1680 PRINT #2:P2$;
1690 IF PCB$="P" THEN 1710
1700 PRINT #2:TAB(12);C2$;"*";
1710 PRINT #2:TAB(24);Q$;TAB(30);ITEMDESC$
1720 TOTC=TOTC+C1
1730 TOTP=TOTP+P1
1740 CTRS=CTRS+1
```

(Continued in next column)

(Continued from previous column)

```
1750 IF CTRS<7 THEN 1780
1760 INPUT "ENTER TO CONTINUE: ":CON$
1770 CTRS=0
1780 NEXT I
1790 PRINT "***TOTALS***"
1800 PRINT TAB(6);
1810 IF PCB$="C" THEN 1870
1820 AMT1=TOTP
1830 GOSUB 2030
1840 TOTP$=AMT2$
1850 PRINT AMT2$;" ";
1860 IF PCB$="P" THEN 1910
1870 AMT1=TOTC
1880 GOSUB 2030
1890 TOTC$=AMT2$
1900 PRINT AMT2$;" *";
1910 PRINT
1920 IF PYN$<>"Y" THEN 2010
1930 PRINT #2:"**********************"
1940 IF PCB$="C" THEN 1960
1950 PRINT #2:TOTP$;
1960 IF PCB$="P" THEN 1980
1970 PRINT #2:TAB(12);TOTC$;"*";
1980 PRINT #2
1990 PRINT #2:"***TOTAL*************"
2000 CLOSE #2
2010 INPUT "ENTER TO CONTINUE: ":CON$
2020 GOTO 330
2030 AMT1$=STR$(AMT1)
2040 LENA=LEN(AMT1$)
2050 ON LENA GOTO 2080,2100,2120,2140,2160,
     2180,2200,2220
2060 AMT2$="ERROR"
2070 RETURN
2080 AMT2$="        .0"&AMT1$
2090 RETURN
2100 AMT2$="        ."&AMT1$
2110 RETURN
```

```
2120 AMT2$="      "&SEG$(AMT1$,1,1)&"."&SEG$
     (AMT1$,2,2)
2130 RETURN
2140 AMT2$="     "&SEG$(AMT1$,1,2)&"."&SEG$
     (AMT1$,3,2)
2150 RETURN
2160 AMT2$="    "&SEG$(AMT1$,1,3)&"."&SEG$
     (AMT1$,4,2)
2170 RETURN
2180 AMT2$="  "&SEG$(AMT1$,1,1)&","&SEG$
     (AMT1$,2,3)&"."&SEG$(AMT1$,5,2)
```

(Continued in next column)

(Continued from previous column)

```
2190 RETURN
2200 AMT2$=" "&SEG$(AMT1$,1,2)&","&SEG$
     (AMT1$,3,3)&"."&SEG$(AMT1$,6,2)
2210 RETURN
2220 AMT2$=SEG$(AMT1$,1,3)&","&SEG$
     (AMT1$,4,3)&"."&SEG$(AMT1$,7,2)
2230 RETURN
2240 END
```

```
JEFF GORDON

      PRICE        COST QTY   ITEM#/DESCRIPTION

    500.00      300.00* 10    0101 ITEM X
    500.00      250.00* 5     0150 ITEM Y
  1,400.00      700.00* 7     0250 ITEM Z
  4,500.00    2,250.00* 15    0350 ITEM XYZ
     50.00       30.00* 1     0101 ITEM X
**********************
  6,950.00    3,530.00*
***TOTAL*************
```

```
JEFF GORDON

      PRICE            QTY   ITEM#/DESCRIPTION

    500.00             10    0101 ITEM X
    500.00             5     0150 ITEM Y
  1,400.00             7     0250 ITEM Z
  4,500.00             15    0350 ITEM XYZ
     50.00             1     0101 ITEM X
**********************
  6,950.00
***TOTAL*************


JEFF GORDON

                  COST QTY   ITEM#/DESCRIPTION

               300.00* 10    0101 ITEM X
               250.00* 5     0150 ITEM Y
               700.00* 7     0250 ITEM Z
             2,250.00* 15    0350 ITEM XYZ
                30.00* 1     0101 ITEM X
**********************
             3,530.00*
***TOTAL*************
```

RULE78

```
100 REM * ************************
110 REM *          RULE78          *
120 REM *                          *
130 REM *      COPYRIGHT 1983      *
140 REM *    DONALD C. KREUTNER    *
150 REM * ************************
160 CALL CLEAR
170 GOTO 200
180 INPUT "ANOTHER (Y/N): ":YN$
190 IF YN$="N" THEN 1080
200 INPUT "PRINTER (Y/N): ":PYN$
210 INPUT "PAUSE (Y/N): ":PAYN$
220 IF PYN$<>"Y" THEN 240
230 OPEN #1:"RS232.BA=1200"
240 INPUT "BORROWER: ":N$
250 INPUT "PRINCIPAL:":P
260 INPUT "FINANCE AMT:":F
270 INPUT "MONTHS OF PAYBACK: ":M
280 IF PYN$<>"Y" THEN 390
290 INPUT "COMPRESSED PRINT (Y/N): ":CYN$
300 IF CYN$<>"Y" THEN 320
310 PRINT #1:CHR$(29)
320 PRINT #1:CHR$(12)
330 PRINT #1:" BORROWER: ";N$;"  (Rule of 78s
    Interest)"
340 PRINT #1:" ORIGINAL PRINCIPAL: ";P;
    "        FINANCE AMT: ";F
350 PRINT #1
360 PRINT #1:" MO";TAB(6);"PRINCIPAL";TAB(23);
    "FIN AMT";TAB(35);"TOT FINAMT";TAB(53);
    "PAYMENT"
370 PRINT #1:" ----------------------------------
    ----------"
380 PRINT #1
390 TOTF=0
400 PAY=(P+F)/M
410 AMT=PAY
```

RULE78 should be a familiar title to the banking community. The basis of the program is this: on an installment loan, the finance amount that is assessed for the principal loaned is divided among the months over which the payments will be made by using the "rule of 78s" formula. The name comes from the sum of months 12, 11, 10, 9, and so on, down to 1. The sum of these digits over a one-year period is 78. And to split the interest over that year, the first month is 12/78ths of the total finance charge, the second is 11/78ths, etc., down to the 12th month, which gets 1/78th of the interest. With this formula you can determine the payoff balance of any loan, since you know how much interest has been earned as of any month.

Sound complicated? Well, there's more. Obviously, most loans are for more than 12 months. To calculate the sum of the months (for example, 12 down to 1, for a one-year loan), the program has only to multiply the number of months times the number of months plus 1, and divide by 2 (12 x 13 divided by 2 = 78).

That's the concept—but you say you couldn't care less? The beauty of RULE78 is that all you have to do is feed the computer these few numbers:

1. The number of months for the loan
2. The total finance charge
3. The original principal, or borrowed amount

```
420 GOSUB 790
430 PAY=AMT
440 FOR I=1 TO M
450 FM=(M+1-I)/((M*(M+1))/2)*F
460 AMT=FM
470 GOSUB 790
480 FM=AMT
490 PRINT "**********MONTH";I
500 IF I<>M THEN 530
510 FM=F-TOTF
520 PAY=P+FM
530 AMT=FM
540 GOSUB 870
550 TABFM=TAB1
560 PRINT "FINANCE AMT";TAB(15+TABFM);FM
570 P=P-PAY+FM
580 AMT=P
590 GOSUB 870
600 TABP=TAB1
610 PRINT "PRINCIPAL";TAB(15+TABP);P
620 TOTF=TOTF+FM
630 AMT=TOTF
640 GOSUB 870
650 TABF=TAB1
660 PRINT "TOTAL FIN AMT";TAB(15+TABF);
    TOTF
670 AMT=PAY
680 GOSUB 870
690 TABPAY=TAB1
700 PRINT "PAYMENT";TAB(15+TABPAY);PAY
710 IF PYN$<>"Y" THEN 730
720 PRINT #1:I;TAB(5+TABP);P;TAB(20+TABFM);
    FM;TAB(35+TABF);TOTF;TAB(50+TABPAY);
    PAY
730 IF PAYN$<>"Y" THEN 750
740 INPUT " ":X$
750 NEXT I
760 IF PYN$<>"Y" THEN 180
770 CLOSE #1
```

(Continued in next column)

From these inputs, the program will do all the number crunching and produce a nice schedule, showing the interest paid each month, the payment, the payoff balance, and the total finance amount.

As in several other programs, you can list the printout on the screen and/or to a printer. You can also choose to have a pause in the display to the screen (lines 200-230). Another interesting routine for editing numbers for dollars and cents takes place in the subroutine in lines 790-1070, which returns a number (TAB1) to TAB based on the length of the number being edited.

(Continued from previous column)

```
780 GOTO 180
790 AMT2=AMT*100
800 AMT3=INT(AMT2)
810 AMT4=AMT3+.5
820 IF AMT4<=AMT2 THEN 850
830 AMT=AMT3/100
840 RETURN
850 AMT=(AMT3+1)/100
860 RETURN
870 TAB1=0
880 IF AMT=0 THEN 1040
890 IF AMT<1 THEN 1060
900 IF AMT<10 THEN 1040
910 IF AMT<100 THEN 1020
920 IF AMT<1000 THEN 1000
930 IF AMT<10000 THEN 980
940 IF AMT<100000 THEN 960
950 RETURN
960 TAB1=1
```

```
 970 RETURN
 980 TAB1=2
 990 RETURN
1000 TAB1=3
1010 RETURN
1020 TAB1=4
1030 RETURN
1040 TAB1=5
1050 RETURN
1060 TAB1=6
1070 RETURN
1080 END
```

```
BORROWER: JOHN JONES  (Rule of 78s Interest)
ORIGINAL PRINCIPAL:  5000          FINANCE AMT:  1210.48
```

MO	PRINCIPAL	FIN AMT	TOT FINAMT	PAYMENT
1	4838.07	96.84	96.84	258.77
2	4672.1	92.8	189.64	258.77
3	4502.1	88.77	278.41	258.77
4	4328.06	84.73	363.14	258.77
5	4149.99	80.7	443.84	258.77
6	3967.88	76.66	520.5	258.77
7	3781.74	72.63	593.13	258.77
8	3591.56	68.59	661.72	258.77
9	3397.35	64.56	726.28	258.77
10	3199.1	60.52	786.8	258.77
11	2996.82	56.49	843.29	258.77
12	2790.5	52.45	895.74	258.77
13	2580.15	48.42	944.16	258.77
14	2365.76	44.38	988.54	258.77
15	2147.34	40.35	1028.89	258.77
16	1924.88	36.31	1065.2	258.77
17	1698.39	32.28	1097.48	258.77
18	1467.86	28.24	1125.72	258.77
19	1233.3	24.21	1149.93	258.77
20	994.7	20.17	1170.1	258.77
21	752.07	16.14	1186.24	258.77
22	505.4	12.1	1198.34	258.77
23	254.7	8.07	1206.41	258.77
24	0	4.07	1210.48	258.77

BORROWER: DON JENNINGS (Rule of 78s Interest)
ORIGINAL PRINCIPAL: 1000 FINANCE AMT: 120

MO	PRINCIPAL	FIN AMT	TOT FINAMT	PAYMENT
1	925.13	18.46	18.46	93.33
2	848.72	16.92	35.38	93.33
3	770.77	15.38	50.76	93.33
4	691.29	13.85	64.61	93.33
5	610.27	12.31	76.92	93.33
6	527.71	10.77	87.69	93.33
7	443.61	9.23	96.92	93.33
8	357.97	7.69	104.61	93.33
9	270.79	6.15	110.76	93.33
10	182.08	4.62	115.38	93.33
11	91.83	3.08	118.46	93.33
12	0	1.54	120	93.37

4

TI-99/4A BASIC Commands, Statements, and Functions

Using TI BASIC efficiently is important for successful operation of your computer. This chapter is designed to help you become more familiar with the commands, statements, and functions of TI BASIC. The *TI-99/4A BASIC Reference Card,* which you received with your system, will also be useful in reminding you of the appropriate commands for the programs you write.

Initially, the "trick" is in understanding the statements. Two books that come with your TI-99/4A, the *User's Reference Guide* and *Beginner's BASIC,* are very helpful. If you are really serious about learning all you can, a thorough reading (and rereading) of both these books is suggested. The TI manuals provided with optional peripherals are also quite good in explaining how to access properly such devices as disk drives, printers, etc.

In the following discussion of commands, statements, and functions, each term is explained briefly, yet clearly. As with the *Reference Card,* all terms are arranged in alphabetical order, but more detailed description is given in this chapter, providing yet another source for learning more about TI BASIC and the 99/4A

Strings and Numeric Expressions

To understand any version of BASIC, a programmer needs to know the difference between strings and numeric variables.

Strings can have alphabetical, numerical, or other characters, as well as combinations of these. But a string expression (even one containing all numbers) cannot have arithmetic operations performed on it. For these operations, you first need to convert the string to a

numeric variable (using the **VAL** function). You also can split strings or join them (using **SEG$** and the **&** in the **LET** statement). But to split or join numeric variables is difficult, if not impossible, without first converting them to strings. Strings can be thought of as expressions that can be contained within quotation marks (not to be added, subtracted, etc.).

Unlike strings, numeric variables are composed entirely of numbers. They can be positive or negative and can include decimal portions. Numeric variables should never be enclosed in quotation marks, and any arithmetic operations can be performed on these variables.

The names of string variables end with a $ (dollar sign), which you can pronounce "string." Thus, A$ is called "A STRING," whereas the numeric variable A is simply called A.

Following are some examples of string and numeric variable values.

Strings	Numeric Variables
1024 S. FIFTH	1024
"KEN RICHARDS"	-1.342
"1024"	0

Commands, Statements, and Functions

In the discussion of each BASIC command, statement, and function, you will notice that every entry is followed by a (C), (S), (C,S), or (F), to indicate whether the BASIC keyword is a command, a statement, both a command and a statement, or a function.

Commands	may be typed in and entered after the BASIC prompt > without a preceding line number. In a BASIC program a line number converts a command to a statement. Commands are executed immediately, as they are entered.
Statements	are used within BASIC programs and follow a line number to cause program execution. Obviously, all programs must be composed of statements.
Functions	are certain special-purpose routines that have been built in to BASIC. They enable the user to make certain calculations or string manipulations that would otherwise require writing more BASIC code.

Following are some examples of commands, statements, and functions.

Commands:

>LIST	Causes a program to be displayed on the screen
>OLD "CS1"	Causes a program to be retrieved from cassette tape drive #1

Statements:

10 PRINT "HELLO"	Displays HELLO on the screen when program line 10 is executed
20 CALL CLEAR	Clears the screen when line 20 is executed

Functions:

>PRINT ABS(-10)	Prints the positive value (absolute value) of -10
10 X=VAL(X$)	Returns the numeric value of the string contained in X$ (which should be a number)

After the discussion of each command, function, or statement, examples are given to enable you to understand better the use of the keyword.

Remember that the manual is always there when you need it!

A

ABS(numeric expression)—returns the absolute value of a number, always producing a positive result.

```
10 A=-10
20 B=100
30 PRINT ABS(A)
40 PRINT ABS(B)
```

These lines result in printing 10 and 100. (F)

ASC(string expression)—gives the ASCII value of the first character in a string. If you want to display a string one character at a time to a certain place on the screen, you can use the **ASC** function in combination with the **SEG$** function to obtain the ASCII value of each character for the length of the string. (To determine its length, use the **LEN** function.) Then you can display a string on the screen using the **CALL HCHAR** statement.

```
10 A$="HELLO THERE"
20 LENA=LEN(A$)
30 ROW=5
40 COL=5
50 FOR I=1 TO LENA
60 COL=COL+1
70 SEGX$=SEG$(A$,I,1)
80 ASCI=ASC(SEGX$)
90 CALL HCHAR(ROW,COL,ASCI)
100 NEXT I
```

In this example, line 20 gets the length of A$ in the numeric variable LENA (which is 11, since A$ contains 11 characters). Then both the row and column variables are set to 5 in lines 30 and 40. What we want to accomplish is done by a **FOR-NEXT** loop in lines 50-100, which increases the variable I from 1 to 11 (the value of LENA) by 1, stepping through each statement between the **FOR** and **NEXT** statements 11 times. Each time through the loop, the variable COL is increased by 1 so that the letters of "HELLO THERE" are moved one at a time to the 6th column, the 7th, the 8th, and on up to the 17th column.

The character displayed each time is the one that corresponds to the ASCII value for the 1st character of A$, the 2nd, the 3rd, and so on, up to the 11th character. The **SEG$** function returns the character starting in the 1st position for a length of one character, then the character in the 2nd position, and on through the character in the 11th position. The **ASC** function turns each character into an ASCII code so that the **CALL HCHAR** statement can display the characters as they are encountered. (F)

ATN(radian expression)—obtains the trigonometric arc tangent. Along with certain other functions, **ATN** is useful to engineers and mathematicians.

```
10 A=ATN(.55)
```

This statement gives the angle in radians whose tangent is .55. (F)

B

BREAK [line list]—halts the program when a line list is not given. With a line list, the program stops executing when the designated lines are reached.

```
10 A=20
20 PRINT A
30 PRINT A*2
>BREAK 20
>RUN
*BREAKPOINT AT 20
>PRINT A
20
>CONTINUE
20
40
** DONE **
```

Notice in these lines that when the breakpoint is found, the prompt > is displayed on the screen. You can then type any BASIC command or statement that can be used as a command. This feature is particularly helpful in debugging a program, since you can stop it at

any moment, before it executes the statement(s) specified, to find out any assigned values as of that moment. (See **TRACE** for another valuable debugging tool.) (F)

BYE—causes all open files to be closed. The program will leave BASIC and return to the TI screen. (C)

C

CALL CHAR(character code, pattern identifier)—enables you to define your own graphics characters. For example, you can define a solid block character by stating

```
10 CALL CHAR(97,"FFFFFFFFFFFFFFFF")
```

This block can then be displayed at row 10, column 15, using the **VCHAR** subprogram, as in

```
20 CALL VCHAR(10,15,97)
```

The pattern identifier is a 16-character string, of which each group of two characters represents the number of filled-in blocks of the left and right sections of each of eight rows. By setting certain blocks "on" and certain blocks "off" within the grid of 64 blocks, you can describe almost any graphics character you need. (C,S)

CALL CLEAR—clears the entire screen, usually at a program startup or when you want to "wipe out" the residue left on the screen from a previous section of the program (such as before returning to a menu screen to ask what action you want to take next).

```
10 CALL CLEAR
20 PRINT "MAIN MENU COMMANDS"
30 PRINT
40 PRINT "R      READ A FILE"
50 PRINT "W      WRITE A FILE"
60 PRINT "L      LIST MEMORY CONTENTS"
70 PRINT "C      CHANGE MEMORY CONTENTS"
80 PRINT "N      NEW (CLEAR MEMORY)"
90 PRINT
100 PRINT
110 INPUT "SELECTION: ":S$
```

These lines clear the screen before "painting" it with the menu offerings in the **PRINT** statements. (C,S)

CALL COLOR(character set, foreground color, background color)—provides you with the capability of designing your own character and background colors. If you are using a color TV or a color monitor, this subprogram can be useful. There are 16 color shades available, with assigned numbers from 1 to 16. The foreground color is the color of the blocks that make

up the character, whereas the background color is the color of the "unused" blocks (those that are "off").

```
10 CALL COLOR(2,3,4)
```

This statement causes characters in the character set 2 (the { through /, from the character set chart in the TI manual) to be displayed as medium green on a light green background. (C,S)

CALL GCHAR(row, column, numeric variable)—allows you to read a character from any position on the screen. You simply specify the row and column and give a numeric value. The subprogram will return the ASCII code for the character at that location.

```
10 CALL GCHAR(5,5,NUM1)
```

This statement puts the ASCII value for the character on the screen at row 5, column 5, into the numeric variable NUM1. (C,S)

CALL HCHAR(row, column, character code [,repetitions])—places the character of the specified ASCII character code at the row and column requested. If the repetitions option is used, **CALL HCHAR** repeats the character horizontally a specified number of times.

```
10 CALL HCHAR(10,6,64,20)
20 A$="A"
30 CALL HCHAR(11,6,ASC(A$),20)
```

In line 10 the statement displays 20 @'s ("at" signs) across the screen, starting at row 10, column 6. Then one row down, 20 A's will be displayed. Notice that you can use **CALL HCHAR** or similar commands requiring the ASCII character code without knowing it (by using the **ASC** function to obtain that code). (C,S)

CALL JOYST(key unit, x return, y return)—enables you to input values interpreted as requests for motion in certain directions. The input comes from moving the joysticks (key units 1 and 2). Using a 3, 4, or 5 with this subprogram will cause no input detection (by the computer) of joystick motion. Depending on the direction the joystick is pointed, the input that is received ranges from -4 to 0 and from 0 to +4 for the x and y return values. Motion to the right for the x return is +4, motion to the left is -4, and no motion left or right is 0. Likewise, for the y return, motion up is +4, motion down is -4, and no motion up or down is 0. From the x and y value returned, you can then move characters around the screen.

Remember that to be able to use the joysticks, you must have the alpha lock key off. Otherwise, you will not be able to move up (or north).

```
100 INPUT "DRAW/JUMP (D/J): ":DJ$
110 CALL CLEAR
120 CALL CHAR(97,"0000003C3C000000")
```

```
130 R=10
140 C=10
150 CALL HCHAR(R,C,97)
160 C1=C
170 R1=R
180 CALL JOYST(1,X,Y)
190 IF X=4 THEN 210
200 IF X=-4 THEN 230
205 GOTO 240
210 C=C+1
220 GOTO 240
230 C=C-1
240 IF Y=4 THEN 290
250 IF Y=-4 THEN 270
260 GOTO 300
270 R=R+1
280 GOTO 300
290 R=R-1
300 IF R<5 THEN 320
310 GOTO 330
320 R=5
330 IF R>20 THEN 350
340 GOTO 360
350 R=20
360 IF C<5 THEN 380
370 GOTO 390
380 C=5
390 IF C>20 THEN 410
400 GOTO 420
410 C=20
420 IF DJ$<>"J" THEN 440
430 CALL HCHAR(R1,C1,32)
440 CALL HCHAR(R,C,97)
450 GOTO 160
```

This joystick program is a fairly elaborate example. Its graphics motion is similar to that in the game TAG, discussed in Chapter 2.

Line 100 asks if you want the small dot defined in the **CALL CHAR** statement in line 120 to "draw" smoothly or to jump around the screen in response to the commands issued by the joystick. Line 110 then clears the screen before beginning. The initial position of the

character is given by the row (R) and column (C) coordinates in lines 130 and 140. Each time a move is made, lines 160 and 170 "remember" where the character was last located.

Then the joystick command and the statements in lines 180-290 compute the new position of the character, depending on which direction the joystick was pointed. Lines 300-410 keep the character within rows 5-20 and columns 5-20. The joystick will not move beyond these limits. Finally, if you ask for jumping, the last character is erased in line 430 by placing a space wherever the character was (row R1, column C1). Then the new position is plotted at line 440 before control loops back to line 160 for the next joystick scan. (C,S)

CALL KEY(key unit, return variable, status variable)—scans the keyboard to determine when a key has been depressed, as with the joystick command. The key unit may be numbered from 0 to 5 for various types of scans, but the console keyboard scan uses 0. The return variable to this type of scan receives as its value the ASCII code of the key depressed.

```
10 CALL KEY(0,A,STATUS)
20 IF STATUS=0 THEN 10
30 PRINT CHR$(A)
40 GOTO 10
```

These lines scan for a key to be hit on the keyboard. If no key is sensed, line 20 branches back to line 10 for another scan, and so on. If a key is depressed, the ASCII value obtained is displayed as a character representation by line 30. (C,S)

CALL SCREEN(color code)—changes the screen's background color by using the same 16-color code numbers for the **CALL COLOR** command. When the screen color is changed, colors of characters already on the screen remain the same (unless they are transparent).

```
10 CALL CLEAR
20 INPUT "KEY 1-16 FOR SCREEN COLOR: ":C
30 IF C>16 THEN 20
40 IF C<1 THEN 20
45 CALL SCREEN(C)
50 INPUT "CHARACTER COLOR: ":CC
60 IF CC>16 THEN 50
70 IF CC<1 THEN 50
80 INPUT "BACKGROUND CHARACTER COLOR: ":BC
90 IF BC>16 THEN 80
100 IF BC<1 THEN 80
110 CALL CLEAR
120 CALL COLOR(3,CC,BC)
130 FOR I=1 TO 5
140 PRINT "CHARACTER: ";I
```

```
150 NEXT I
160 INPUT "RETURN TO CONTINUE/E TO END":E$
170 IF E$<>"E" THEN 10
180 END
```

This subprogram illustrates the use of many different screen colors, as well as the ability to control both the foreground and background colors of the characters 0 to 7 (subset 3). After keying the desired screen, character, and background colors, a count from 1 to 5 shows these color combinations. By keying **RETURN**, you can then run the program again. (C,S)

CALL SOUND(duration 1, frequency 1, volume 1; duration 2, frequency 2, volume 2; . . . volume 4)—causes the generation of tones or noises at the desired length, pitch, and volume. Within a command, up to three tones and one noise may be specified to be activated simultaneously. Thus, you can blend musical notes or noise sounds to make your own "music."

```
10 CALL CLEAR
20 INPUT "DURATION (-4250 TO -1, 1 TO 4250): ":D
30 INPUT "FREQUENCY (110 TO 44733, -8 TO -1): ":F
40 INPUT "VOLUME (0, LOUD TO 30, SOFT): ":V
50 CALL SOUND(D,F,V)
60 GOTO 10
```

A negative duration causes a previous sound to stop and a new one to begin at once. Duration is in milliseconds (1000 milliseconds = one second).

Frequency is measured in Hertz (Hz.), from a very low-pitched 110 Hz. to an inaudible 44,733 Hz. A negative value for frequency causes a noise to be created instead of a tone. Volume ranges from a loud 0 to a soft 30. (C,S)

CALL VCHAR(row, column, character code, [, repetitions])—repeats a character vertically instead of horizontally, for the optional number of repetitions specified. (**CALL VCHAR** is very similar to **CALL HCHAR.**)

```
10 CALL CLEAR
20 A$="A"
30 CALL HCHAR(11,6,ASC(A$),20)
40 CALL VCHAR(2,16,ASC(A$),20)
50 INPUT X$
```

Here the lines will put 20 A's across the screen from the 11th row and the 6th column, and then 20 A's down the screen from the 2nd row and the 16th column. (C,S)

CHR$(numeric expression)—returns the string character, when given an ASCII code, which that code represents.

```
10 CALL CLEAR
20 INPUT "KEY AN ASCII CODE (33-126): ":C
30 IF C<33 THEN 20
40 IF C>126 THEN 20
50 PRINT "ASCII ";C;" IS: ";CHR$(C)
60 INPUT "MORE (Y/N): ":YN$
70 IF YN$="Y" THEN 10
80 END
```

The **CHR$** statement in line 50 obtains the string value for the ASCII code keyed, telling you what number the computer uses to keep track of different characters in its memory. (F)

CLOSE #file number [:**DELETE**]—closes, or disassociates from the program, a file that has been opened. After a file is closed, you can reuse its file number in opening another file. (See **OPEN**.) The **DELETE** option is applicable only to certain devices, such as the disk drive, in deleting the file when it is closed.

```
10 OPEN #1:"RS232.BA=1200"
20 PRINT #1:"HELLO"
30 CLOSE #1
40 OPEN #2:"DSK1.FILE1"
50 PRINT #2:"HELLO"
60 CLOSE #2:DELETE
```

The **CLOSE** statement in line 30 makes the printer file opened in line 10 no longer available to the program. Thus, if you add a line 40 (PRINT #1:"GOODBYE"), the program will end in an error condition. (C,S)

CONTINUE or **CON**—resumes the execution of a program after a breakpoint has been encountered. (See also **BREAK**.) (C)

```
10 PRINT "LINE 10"
20 PRINT "LINE 20"
30 PRINT "LINE 30"

>BREAK 20
>RUN
LINE 10

* BREAKPOINT AT 20
>CONTINUE
LINE 20
LINE 30
```

COS(radian expression)—returns the trigonometric cosine of the radian expression, an angle in radians. To get the cosine of an angle in degrees, you first need to multiply the angle in degrees by PI/180, which is .01745329251994, to get the radians for that angle. (F)

```
10 CALL CLEAR
20 INPUT "COSINE OF DEGREES OR RADIANS (D/R): ":DR$
30 INPUT "VALUE OF ANGLE: ":DR
40 IF DR$="R" THEN 60
50 DR=DR*.01745329251994
60 PRINT "COSINE OF ";DR;" IS ";COS(DR)
```

D

DATA data list—stores numeric or string data within a program's statements so that you can use the same data each time you run the program.

```
10 CALL CLEAR
20 DATA 10,11,15,32,9,12,10,-99
30 READ A
40 IF A<0 THEN 70
50 PRINT A
60 GOTO 30
70 END
```

Line 20 provides the numeric program data to be read in by line 30. To end the program without a DATA ERROR message, line 40 will branch to the end of the program (line 70) if a negative number is read (-99). Line 50 prints the number read on the screen, and the program loops back to read the next data element from line 60. (S).

DEF function name[(parameter)]=expression—defines whatever functions you want in your own programs. If you include an optional parameter, it can be used to evaluate the value returned by the function.

```
10 DEF PRICE=COST*1.7
20 INPUT "COST: ":COST
30 PRINT "PRICE: ";PRICE
40 GOTO 20
```

DEF is not really necessary in a program like this since line 30 can just as easily state PRINT "PRICE: ";COST*1.7. (S)

DELETE filename or program name—removes a file or program from disk. (C)

```
>DELETE "DSK1.PROG1"
```

DIM {array name (integer 1, integer 2, integer 3)}—reserves space for arrays of 1, 2, or 3 dimensions. The number of values following the array name in parentheses tells whether the array is 1-, 2-, or 3-dimensional.

```
10 DIM A(10)
```

Here the statement reserves space for an array of 11 numbers, for without the statement **OPTION BASE** 1, the subscript 0 is also counted. (See **OPTION BASE.**)

```
10 DIM A$(3,3)
```

This statement dimensions a 2-dimensional array. Such an array can store the names of up to three family members for each of three families. For example, A$(1,1), A$(1,2), and A$(1,3) each would contain a different name for a member from family #1. (C,S)

DISPLAY print list—prints elements only to the screen, not to files, as in the **PRINT** #file number command. (C,S)

```
10 PRINT "ENTER YOUR NAME"
```

is identical to

```
10 DISPLAY "ENTER YOUR NAME"
```

E

EDIT line number—allows you to enter the edit mode on a line number in order to insert, delete, or erase characters in a line while you are changing a BASIC program. You can also enter the edit mode by keying the line number followed by the up-arrow function (↑) or the down-arrow function (↓). (C)

END—terminates the execution of a program. The last statement is usually **END**, to which you branch to stop the program. However, **END** may be located elsewhere within the program, if desired. (C,S)

EOF(numeric expression)—checks the condition of the input file before trying to input data, thus preventing an error condition from occurring if there is no data to input. This **EOF** function works with a normal sequential disk file, not with a cassette file or a relative disk file.

```
10 OPEN #1:"DSK1.FILEX"
20 IF EOF(1) THEN 50
30 INPUT #1:X$
40 GOTO 20
50 CLOSE #1
60 END
```

Here line 20 will branch to end the program if there is no more data to read. With **EOF** you do not need a special identifying record to indicate that you are at the end of the file (like "END" or "99999"). (F)

EXP(numeric expression)—provides the inverse value of the natural **LOG** function. The value returned represents e (that is, 2.718281828) to the power of the argument in parentheses. **EXP** is used in mathematical formulas and applications.

```
10 PRINT EXP(9)
```

This statement prints out the value of e to the 9th power. (F)

F

FOR control variable=initial value **TO** limit [**STEP** increment]—executes the statements that follow **FOR**, up to the **NEXT** control variable statement, which is located farther down in the program. This "loop" will be executed counting from the initial control variable until the limit is reached. Usually, the control variable is incremented by one, but this increment may vary by stating a **STEP** option. With **STEP** 5, for example, you can have the control variable incremented by 5 each time the loop is executed. After the limit has been reached, control passes to the statement after the **NEXT** statement. This statement is similar to the DO-LOOP of FORTRAN.

```
10 CALL CLEAR
20 INPUT "DISPLAY COUNT (Y/N): ":YN$
30 INPUT "HOW HIGH TO COUNT: ":L
40 INPUT "COUNT BY: ":S
50 INPUT "COUNT FROM: ":C
60 CALL CLEAR
70 FOR I=C TO L STEP S
80 IF YN$<>"Y" THEN 100
90 PRINT I
100 NEXT I
110 INPUT "MORE (Y/N): ":YN$
120 IF YN$="Y" THEN 10
130 END
```

This program illustrates the versatility of the **FOR-NEXT** loop. With it, you can make the TI-99/4A display each number as it counts. You will be amazed at how much the speed of the computer is held back because it has to "show" you what it is doing as it does it. The speed of the computer is much more evident when you tell it not to display the count. You see, the baud rate (or speed at which the screen is filled with characters) is much slower than the speed at which the computer is capable of working. Obviously, one of the reasons for this discrepancy

is that people must be able to read information as it is displayed on the screen. If material went by too fast, the machine would be of little use.

You can also key the limit to which you want the machine to count, as well as the number from which to start. For example, you can tell the computer to count from 1 to 1000. You can also give the computer a step of -1 and ask for a count to 1 from a starting number of 1000, causing a count backwards. Or you can have the computer count from 10 to 2000 by 10s, 20s, etc. (S)

G

GOSUB line number—enables you to "perform" (in COBOL terms) or "call" (in FORTRAN terms) a subroutine that can be done easily, over and over again, from different sections of a program. You can, for instance, load some numbers from one section of a program into certain variables, **GOSUB** to a certain line number, then return to the statement following the **GOSUB**. You can then use the variables, as they have been changed by the subroutine, in succeeding statements. Later, you can load any other numbers into the same variables, **GOSUB** to the same line number (from a completely different section of the program), and return, after encountering the **RETURN** statement, to the next statement after the **GOSUB**.

```
10 INPUT X
20 GOSUB 100
30 PRINT X
40 X=X*2
50 GOSUB 100
60 PRINT X
70 GOTO 10
100 X=X/2
110 RETURN
```

Here, when you key a number (X) in line 10, the **GOSUB** statement in line 20 transfers control to line 100, where X is divided by 2. When the **RETURN** statement is encountered in line 110, control passes back to line 30, where X is printed. (If you key 10 for X, the subroutine will print 5.) Line 40 then doubles X (back to its original value). But the **GOSUB** in line 50 again branches to line 110, where X is again divided by 2, and the value of X is once again printed in line 60. Then the **GOTO** in line 70 takes you back for another number to be input. Most subroutines have many more statements than the example above. (S)

GOTO line number—goes to the line number specified. Also called the unconditional branch, **GOTO** is one of the most frequently used BASIC statements. The **GOTO** (or **GO TO**) statement, in combination with the conditional branch **IF-THEN-ELSE**, was used to develop the "stored program" concept in earlier generations of computers.

```
10 INPUT "Number 1: ":N1
20 INPUT "Number 2: ":N2
30 INPUT "Add or Subtract (A/S): ":AS$
40 IF AS$="A" THEN 70
50 PRINT N1-N2
60 GOTO 10
70 PRINT N1+N2
80 GOTO 10
```

The conditional branch in line 40 takes the program to line 70 if the user selected to add the two numbers. (Otherwise, the program goes to line 60 after printing the subtraction in line 50.) The **GOTO** statements of lines 60 and 80 both unconditionally branch to line 10. (S)

I

IF relational or numeric expression **THEN** line number [**ELSE** line number 2] —executes the statement following **THEN** if the condition specified is true. If it is not true, control passes to the next statement, unless the optional **ELSE** clause is used to transfer control to line number 2. **IF-THEN** is the conditional branch that helped make the computer one of the most important innovations of the 20th century.

```
10 INPUT N
20 IF N>0 THEN 60
30 IF N<0 THEN 80
40 PRINT "THE NUMBER WAS ZERO"
50 GOTO 100
60 PRINT N;"IS GREATER THAN ZERO"
70 GOTO 100
80 PRINT N;"IS LESS THAN ZERO"
100 INPUT "CONTINUE (Y/N): ":YN$
110 IF YN$<>"N" THEN 10
120 END
```

Here, after you key a number in line 10, line 20 passes control to line 60 if the number keyed is greater than 0; line 30 branches to line 80 if the number keyed is less than 0. Otherwise, the number has to be 0, and the program "falls through" to line 40, which is the next sequential statement. Another conditional branch evaluates the alphabetical string you answered in line 100. In line 110 the program causes any answer not equal to "N" to branch back to line 10. Otherwise, the program again "falls through" and ends at line 120. (S)

INPUT [input prompt:] variable list—causes program execution to pause momentarily until you enter data from the keyboard and press the **ENTER** key. Along with the **CALL KEY**

statement, **INPUT** is used to provide the program with responsive input from the keyboard. If more than one value is requested by a single **INPUT** statement, you can enter all of them separated by commas. Generally, it is better to ask for only one piece of data (or field) at a time from any **INPUT** statement.

```
10 INPUT X
20 INPUT "NOW ENTER Y: ":Y
30 PRINT X,Y
```

It is also better to include an input prompt, as in line 20, to enable the person keying the data to know what needs to be keyed. In line 10 the only prompt to input data will be a question mark. (S)

INPUT #file number [,REC record number]: variable list—inputs data elements (either numeric or string variables) from a file that has been opened for input. The REC option is valid only when inputting data from a relative disk file. With this file, records can be accessed by record number without having to read the records sequentially (one after the other).

You can **INPUT** one or more variables at a time from a file, but usually you will **INPUT** back the same number of variables that you "wrote" to that file, using the **PRINT** #file number statement. If you use fewer variables than were "**PRINT**ed" to the file, the remaining fields or variables are discarded (unless the **INPUT** statement ends with a trailing comma—a case of "pending input" and not the normal procedure).

```
5 INPUT "FILENAME: ":F$
10 OPEN #1:F$,OUTPUT,INTERNAL,FIXED 128
20 INPUT A,B,C
30 PRINT #1:A,B,C
40 IF A=-999 THEN 60
50 GOTO 20
60 CLOSE #1
70 OPEN #1:F$,INPUT,INTERNAL,FIXED 128
80 INPUT #1:A,B,C
90 PRINT A;B;C
100 IF A=-999 THEN 120
110 GOTO 80
120 CLOSE #1
130 END
```

These lines illustrate several important points to remember. First, notice that once a file number has been **CLOSE**d, it can be used again within the same program (line 70 reopens file #1 for input). However, using a different file # for the second input sequence of the program is also a possibility (and perhaps a better alternative for program readability). If you

key **CS1** in line 5 for the file name, tape drive 1 will be used. If you key **DSK1.FILEX**, that disk file will be used.

Second, the -999 value for A causes the writing to the output file to stop (line 40). Reading that same value on input from the file also causes the input to stop (line 100).

Finally, notice that each input from file #1 in line 80 reads the same three variables from a record as those written out in the **PRINT** statement of line 30. (S)

INT(numeric expression)—gives the largest integer of the numeric expression in parentheses, without exceeding the number itself. In other words, **INT** will truncate any fractional or decimal portion of the number, leaving an integer.

```
10 A=10.43521
20 PRINT INT(A)
30 X=INT(A)
40 PRINT X
```

In this simple example, line 20 prints 10, the truncated integer of the variable A in line 10. Line 40 also prints 10, since the variable X has been set to the value of the integer of A in line 30. (F)

L

LEN(string expression)—determines quickly the length of any string that has been keyed in, read in from **DATA** statements, or input from a file. With this built-in function, you can easily use some other BASIC statements that require you to know the length of a string in order to perform certain manipulations on it.

```
10 INPUT X$
20 LENX=LEN(X$)
30 IF LENX>3 THEN 60
40 PRINT "ERROR... MUST BE AT LEAST 4 CHARACTERS, AND FIRST 3 CHARS
     MUST BE ACCT#"
50 GOTO 10
60 ACCT$=SEG$(X$,1,3)
70 DESC$=SEG$(X$,4,LENX-3)
80 PRINT ACCT$
90 PRINT DESC$
100 GOTO 10
```

In this miniprogram you must input a string of at least four characters in length in line 10. If you do not, you are told to redo it. The first three characters represent the account #. The description of the account is in the 4th position to the end of the string X$. Lines 60 and 70 use the **SEG$** function to split the larger X$ into two segments. Notice that in line 70 the

description begins in position 4 of X$ for a length of: the length of X$ minus 3 (the length of the account #). (F)

[**LET**] numeric variable=numeric expression
string variable=string expression
—assigns the value of a numeric or string expression to the variable to the left of the equal sign. Usually, the optional **LET** at the beginning of the command is left off. You can set variables to numeric constants and fixed strings, or assign keyed or calculated values to variables with the **LET** command.

```
10 A=10
20 B=A*2
30 INPUT X$
40 Y$=X$&STR$(A)
50 PRINT Y$
60 GOTO 30
```

Here line 10 assigns the value 10 to the numeric variable A, whereas B becomes 20 by the statement in line 20 (the value of 10 times 2). After you key a string in line 30, Y$ is assigned the string value of whatever you keyed (X$) joined to the string value of A. Thus, if you key "MONSTER," Y$ will have the value "MONSTER10." (C,S)

LIST line number
line number 1 - line number 2
- line number
line number -
—lists all or part of a program in memory on the screen and (optionally) to other devices, such as a printer, a disk file, or a cassette file. Be aware that a listing to disk or cassette cannot be reloaded back into memory with the **OLD** command. Instead, the file can be used as data only, to be read by a BASIC program.

>LIST
causes a listing of the entire program to the screen (stops when the **CLEAR** function is keyed)

>LIST 10-100
lists from lines 10 to 100 only

>LIST -200
lists from the beginning of the program to line 200

>LIST 200-
lists from line 200 to the end of the program

```
>LIST "RS232.BA=1200":10-100
```
lists from lines 10 to 100 on a printer

```
>LIST "DSK1.PROG01"
```
lists the entire program to a disk file called DSK1.PROG01

LOG(numeric expression)—provides the natural logarithm function, or the inverse of the **EXP** function. **LOG** is generally used for mathematics and engineering applications. (F)

```
10 PRINT LOG(3)
```

N

NEW—clears the screen and erases whatever program lines are in memory, in preparation for keying in a new program. **NEW** returns you to the point at which you turned on your machine and requested BASIC. (C)

```
>NEW
```

NEXT control variable—must be paired with a **FOR** statement. (See **FOR**.) If you fail to end a **FOR** statement with a **NEXT** statement later on in the program, the program will not run because of a **FOR-NEXT** error condition.

```
10 FOR I=1 TO 100
20 PRINT I
30 NEXT I
```

Notice that the **NEXT** statement in line 30 must possess the same control variable (I) as the **FOR** statement in line 10. (S)

NUMBER or **NUM** (initial line, increment)—allows you to enter lines for a BASIC program without keying the line numbers. They will be generated automatically while you are in the number mode. To terminate this mode, you need only to key **ENTER** immediately after a generated line number.

If you do not specify an initial line number and increment, the numbers generated will be from line 100 and in increments of 10. (C)

```
>NEW
>NUM
100 INPUT A
110 PRINT A
120 GOTO 100
130                (hit ENTER to exit number mode)
```

O

OLD file name—loads a program into memory from a cassette tape file, a disk file, or other storage device. (C)

```
>OLD CS1          loads a program from tape
>OLD DSK1.PROGX loads the program file DSK1.PROGX
```

ON numeric expression **GOSUB** line number list—passes control to the line number in the line list for a subroutine. Determining which line number to use is based on whether the integer value of the numeric expression is 1, 2, 3, etc. If it is 1, control passes to the subroutine in the 1st line number of the list. If 2, control transfers to the line number of the second subroutine, and so on. After the subroutine encounters a **RETURN** statement, control returns to the next statement after the **ON-GOSUB** statement. (S)

```
10 CALL CLEAR
20 PRINT "1=SELECTION 1"
30 PRINT "2=SELECTION 2"
40 PRINT "3=SELECTION 3"
50 INPUT S
60 ON S GOSUB 80,100,120
70 GOTO 10
80 PRINT "THIS IS SELECTION 1"
90 RETURN
100 PRINT "THIS IS SELECTION 2"
110 RETURN
120 PRINT "THIS IS SELECTION 3"
130 RETURN
```

ON numeric expression **GOTO** line number list—passes control to the line number, based on the integer value of the numeric expression. When this branching occurs, an unconditional **GOTO** branch does not return to the next line after the **ON-GOTO**. As with **ON-GOSUB**, if the value of the numeric expression is 1, 2, 3, etc., control passes to line number 1, 2, 3, and so on.

```
10 CALL CLEAR
20 PRINT "SELECTION 1"
30 PRINT "SELECTION 2"
40 PRINT "SELECTION 3"
50 INPUT S
60 ON S GOTO 80,100,120
70 GOTO 10
80 PRINT "SELECTION 1"
```

```
90 STOP
100 PRINT "SELECTION 2"
110 STOP
120 PRINT "SELECTION 3"
130 END
```

Notice that in this example, after the branch to lines 80, 100, or 120, the program ends because of the **STOP** and **END** statements. If, however, the **ON-GOTO**s branched to quite different locations within a larger program, each of the program segments could be quite long and could then branch conditionally or unconditionally to other parts of the program. (S)

OPEN #file number:filename (file organization, file type, open mode, record type)—prepares a file on an external storage device (such as a tape or disk) for use by a BASIC program. The program must first check to see whether there is such a file before the program can actually use the file. In the **OPEN** statement, then, you describe to the computer the file's characteristics in order for the computer to verify that everything is correct and to proceed with the rest of the program. If a problem exists, you are given an I/O ERROR message when you try to open the file.

The file number must be an integer between 1 and 255. Usually, you use the lower numbers because they are easier to keep track of when using the files.

Filename means the actual name that identifies, for the program, where the file resides. For example, "CS1" and "CS2" refer to cassette tape drives 1 and 2, whereas "DSK1.ANYNAME" refers to the file on disk drive 1 with the name ANYNAME.

File organization is either *sequential* or *relative*. Sequential is the only valid designator for tape files, but disk files can be accessed either sequentially (that is, one record following another, from the first record on) or randomly. With random access, the disk drive can access immediately any record within that file by the record number of the record desired.

File types are *display* and *internal*. Internal is usually best to use since the data is recorded in the file in the same internal format in which the computer uses the data. Display refers to the format in which a person reads data.

Open modes are *input, output, update,* or *append*. Obviously, input and output refer to reading from or writing to a file. Update means you can read or write. With append you can output or add new records to the end of an existing file.

Record types are *fixed* or *variable*. Fixed records have a set record length, which you may specify. Variable records vary in length depending on the size of the data being written. With variable records you can specify the maximum length to be allowed. (C,S)

OPTION BASE 0 or 1—determines whether arrays will be allowed to have a subscript of 0. If you specify 0, your arrays will, in fact, have one more subscripted element than the number you specify.

```
10 OPTION BASE 0
20 DIM A(20),A$(20)
```

Here A and A$ will actually have 21 elements, from subscript 0 through 20. If you don't actually intend to use subscript 0, you are wasting a little memory unless you use **OPTION BASE** 1, which eliminates the 0 subscripts. Finally, if you specify neither 0 nor 1, the **OPTION BASE** 0 will be the default in a program using arrays. (S)

P

POS(string1, string2, numeric expression)—searches through a string for a particular substring that you are trying to find. In a file of names, for example, you may want to search for any name that has, as a part of it, "DON" or "JOHN."

String2 contains the substring to search for in string1, starting in the position of string1, as specified in numeric expression.

```
10 A$="JOHN ADAMS"
20 P=POS(A$,"JOHN",1)
30 PRINT P
```

In this example the computer prints 1, indicating it has found "JOHN" in A$ at position 1. (F)

PRINT print list
#file number [,REC record number]:print list
—"prints" data in the print list to the screen. Without the #file number option this command is exactly like the **DISPLAY** statement.

With the #file number option the program writes the data to an accessory device (for example, disk, tape, or printer). Again, note that when you write to files, internal format is much easier for the computer to handle. The commas separating data items are automatically taken care of with internal format, but not with display format.

```
100 OPEN #1:"CS1",INTERNAL,OUTPUT,FIXED 128
110 A=1023
120 A$="HELLO"
130 PRINT #1:A,A$
```

The statement in line 130 "prints" two variables to a record in the file opened as cassette tape unit 1. These two variables can then be **INPUT** back from that file.

The REC option is used to write or rewrite data to a specific record number within a random disk file only. (C,S)

R

RANDOMIZE (seed)—resets the random number generator to provide an unpredictable random number when using the **RND** function. If a seed number (an integer) is used, you will obtain the same predictable random numbers every time you use that same seed.

```
10 RANDOMIZE
```

This statement causes a random number selection whenever the **RND** function is used. (C,S)

READ variable list—"reads in" variables from **DATA** statements to the variables listed in the **READ** statement.

```
10 DATA "Bacon, Matt"
20 DATA "Sparkman, Mike"
30 DATA "LaDuke, Steve"
40 DATA "Suddeth, Pat"
50 DATA "Nealy, Carl"
60 DATA "Zangmaster, John"
70 DATA "Stewart, Bobby"
80 DATA "Stokes, David"
90 DATA "Howard, Todd"
100 DATA "Rush, Todd"
110 DATA "Brown, Art"
115 DATA "END"
120 READ NAME$
130 IF NAME$="END" THEN 160
140 PRINT NAME$
150 GOTO 120
160 END
```

In this illustration the **READ** statement reads in names from the **DATA** statements until a name that contains "END" is found. If it is not found, an OUT OF DATA error message occurs when line 120 tries to read another variable from the **DATA** statements and finds they have all been used. (S)

REM—allows you to put in your programs explanatory **REM**arks that have no effect on actual program execution. When the program is **RUN**, each **REM** statement is actually skipped over for the next statement. (C,S)

```
10 REM THIS IS A TYPICAL REMARK STATEMENT SERIES
20 REM IN THIS PROGRAM, YOU WILL INPUT A VALUE
```

```
30 REM TO WHICH THE PROGRAM WILL COUNT
40 INPUT "VALUE TO COUNT TO: ":V
50 FOR I=1 TO V
60 PRINT I
70 NEXT I
80 END
```

RES or **RESEQUENCE** [initial line] [, increment]—allows you to renumber the program numbers for your BASIC program so that the numbers are in even increments, and gaps appear in your program for inserting new statements.

```
>RES 100,100
```

All lines within the program will be resequenced starting from line 100 and incrementing by 100 if no optional values are specified. Otherwise, the program will be renumbered starting at the initial line specified and in increments of the specified increment value. If you need to insert up to 100 lines between two lines, this command allows you to do just that.

After adding the lines, you can use the **RES** command to renumber the program in even intervals. (C)

RESTORE #file number [, REC record number]
[line number]
—tells the program from which line to start reading **DATA** statements for a **READ** statement. If you use **RESTORE** only, the data pointer will be reset to the first **DATA** statement within the program, or you can specify the line number to which you want to **RESTORE**.

When used for sequential files, the **RESTORE** statement resets a file to the first record. The REC record number option is valid only for relative files.

```
10 DATA 10,4,5,3,-999
20 READ A
25 PRINT A
30 RESTORE
40 READ B
45 PRINT B
   . . .
```

In this program line 20 will **READ** the value 10 into A. Then line 30 **RESTORES** the data pointer back to the beginning so that line 40 also **READ**s 10 into B. (C,S)

RETURN—passes control from the subroutine being performed to the line after the **GOSUB** statement. (See also **GOSUB**.) (S)

RND—generates random numbers when used with the **RANDOMIZE** statement. WITH **RND**, you can make choices that are arbitrary (by chance).

```
10 RANDOMIZE
20 INPUT "READY? (HIT ENTER): ":R$
30 VALUE=INT(6*RND)+1
40 PRINT VALUE
50 GOTO 20
```

These statements, for example, simulate a throw of a six-sided die.

When you select a random number between 1 and some ending number, the result is the integer value of (the maximum value times a random number generated greater than 0 and less than 1) + 1. (F)

RUN [line number]—starts program execution at the lowest line number when one is not given. If a line number is given, the program executes from the one specified.

```
>RUN

>RUN 50
```

The second command starts execution at line 50. (C)

S

SAVE file name—saves a copy of the program in memory to the device specified in the file name. (C)

>**SAVE** CS1	saves to cassette drive 1
>**SAVE** DSK1.NAME1	saves to disk file named "DSK1.NAME1"

**SEG$**(string expression, position, length)—permits you to create a small substring that is part of a larger string. For example, if you want B$ to contain the characters in A$ starting in position 1 for a length of 4, the following statement accomplishes that:

```
200 B$=SEG$(A$,1,4)
```

Of course, A$ must have a length of at least four characters. (A$ should be much longer, or there is no need to use the **SEG$** function.) (F)

SGN(numeric expression)—returns -1 if the numeric expression is less than 0. If the expression is 0, then 0 is returned. if the expression is positive, 1 is returned. (F)

```
10 A=10
20 PRINT SGN(A)
30 B=-20
40 PRINT SGN(B)
```

```
>RUN
1
-1

** DONE **
```

SIN(radian expression)—returns the trigomometric sine of the angle in parentheses, where the angle is expressed as radians. To convert an angle in degrees to radians, you need to multiply by PI and divide by 180 (or just multiply the angle in degrees by .01745329251944). (F)

The sine, then, of any angle in degrees is determined in the following lines:

```
10 CALL CLEAR
20 INPUT "ANGLE IN DEGREES: ":D
30 S=SIN(D*.01745329251944)
40 PRINT S
50 GOTO 20
```

SQR(numeric expression)—gives the square root of the numeric expression.

```
10 FOR I=1 TO 100
20 PRINT I,SQR(I)
30 NEXT I
```

These lines give you a square root table for all integers between 1 and 100. (F)

STOP—stops a program at various points. **STOP** is similar to the **END** statement, except that **END** is usually the last statement in a program. Ordinarily, there is only one **END**, but any number of **STOP** statements may appear in a program, although this recurrence is not necessarily the best programming procedure. (C,S)

```
200 INPUT "NUMBER BETWEEN 1 AND 10:":N
210 IF N<1 THEN 240
220 IF N>10 THEN 240
230 GOTO 250
240 STOP
250 PRINT "OK YOUR NUMBER WAS WITHIN LIMITS"
260 GOTO 200
```

STR$**(numeric expression)—changes a numeric value to a string value, allowing you to do string manipulations with it (such as determine its length or do a **SEG$). (F)

```
10 A=10
20 A$=STR$(A)
```

```
30 PRINT LEN(A$)
40 PRINT SEG$(A$,1,1)

>RUN
2
1

** DONE **
```

T

TAB(numeric expression)—causes the **PRINT** or **DISPLAY** statements to tab to a certain column before printing the next item. The line

```
10 PRINT "HELLO";TAB(10);"THERE"
```

results in

```
HELLO     THERE
```

TAB can be used for either output to the screen or formatting a printout to a printer. (F)

TAN(radian expression)—provides the trigonometric tangent function of the angle expressed in radians. **TAN** is similar to the **SIN** and **COS** functions. To evaluate an angle in degrees, use the same method to convert degrees to radians as was used with the **SIN** function. (F)

TRACE—displays on the screen each line number as it is executed when this function is turned on before running your program. If your program does not work properly, you may find the **TRACE** command useful. With it, you may discover that a certain branch in the program is not taking place because of an error in your logic. (The **UNTRACE** command turns off the **TRACE**.) Remember also that **BREAK** is another valuable debugging tool.

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I

>TRACE
>RUN
```

With **TRACE** turned on, this program will print 10, then 20 and 30, 20 and 30, over and over, 10 times, as the loop is executed. (C,S)

U

UNBREAK [line list]—removes all breakpoints set by the **BREAK** statement when no line list is given. With the optional line list, only those specified are removed. (C,S)

```
>UNBREAK
```

UNTRACE—turns off the **TRACE** feature. (C,S)

```
>UNTRACE
```

V

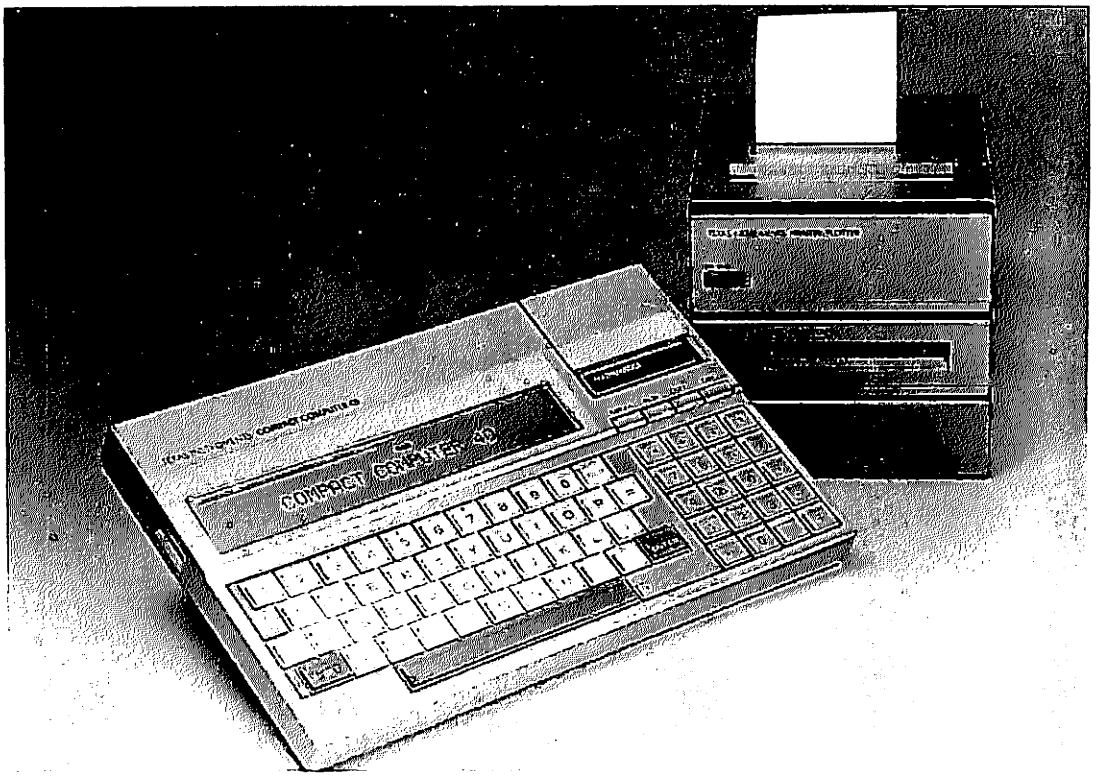
VAL(string expression)—converts a string expression to a numeric expression. **VAL** is the reverse of the **STR$** function, since **STR$** converts a numeric expression to a string expression.

```
10 A$="109"
20 A=VAL(A$)
30 PRINT A;A*2
```

Here the value 109 is put into the numeric variable A, and line 30 prints 109 and 218. (F)

Appendix
New TI Products

Texas Instruments has announced several new products that will soon be available. Many of these products are based on the HEX-BUS™ concept, a means of "daisy-chaining" many devices in a series, from one to another, and from device to computer, using small cables. These peripherals were developed for TI's new Compact Computer, the CC-40. The computer, the Printer/Plotter, the Wafertape™ Drive, and the RS-232 Interface are pictured below.



The portable TI CC-40 Compact Computer

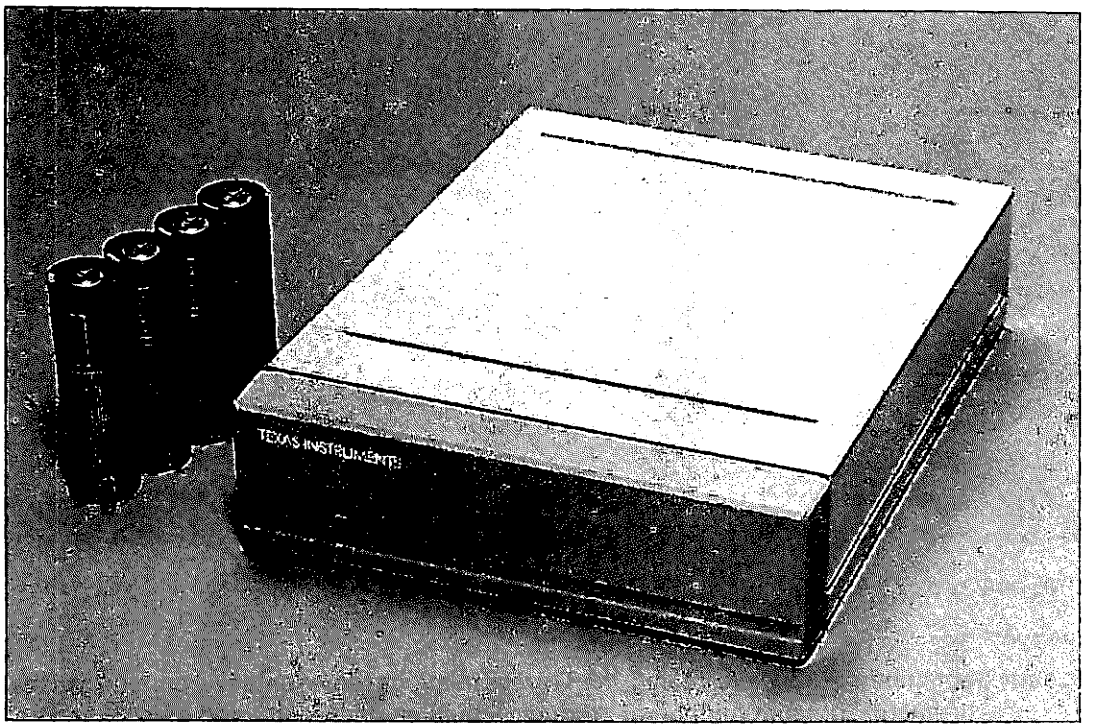
The CC-40 is battery powered (AA size) and can run for up to 200 hours without changing batteries. The one-line display can scroll backwards and forwards through a BASIC program when the up (↑) or down (↓) arrow keys are pressed. This easy-to-use computer is designed for portable use.

Perhaps the most exciting thing about TI's new technology is the development of the compact peripherals for the CC-40. Because these devices will also be portable, an entire computer system can be carried in a large camera case.

These same peripherals can be used on the TI-99/4A by attaching a HEX-BUS interface adapter at the side of the console. The home computer owner will then have a choice between the very fast disk system that is currently available, and the new, slower, but still quite fast, "wafertape" storage. This tape drive can store over 48,000 bytes of information at a transfer rate of 8,000 bits per second. The Printer/Plotter prints on standard, 2 1/4-inch adding machine paper, available in four colors. Ten different type sizes can be used, with up to 36 characters per line. An RS-232 Interface is also available for connecting to a standard-sized printer, or to a modem by this same HEX-BUS interface system. The HEX-BUS Interface is pictured below.

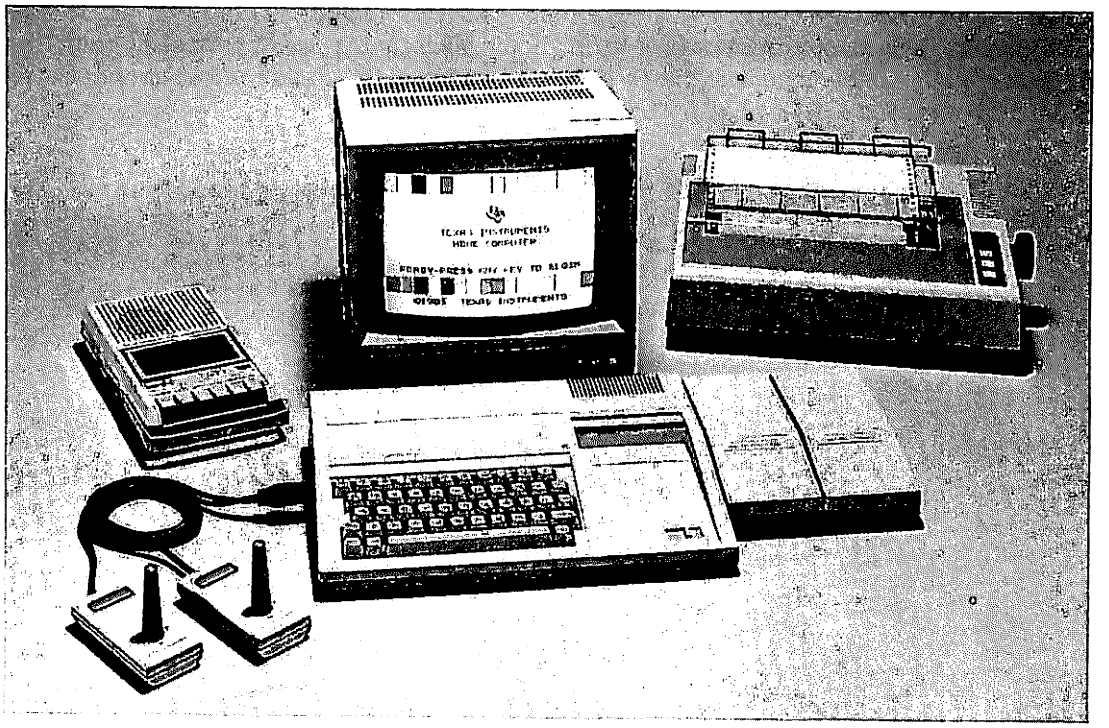
The HEX-BUS Interface for the TI-99/4A

Other products designed for the HEX-BUS Interface system include a video interface for the CC-40, enabling a display of 24 lines of 40 characters each; and a new, low-cost modem. The modem, which can be used with the TI-99/4A, is 300 baud, full duplex, direct connect, and BELL 103 compatible. Below is an illustration.



Texas Instruments' new HEX-BUS modem

Two other notable changes are taking place in the new TI-99/4A Home Computer. The first is a cosmetic change. Since Texas Instruments has found through research that users prefer lighter colors in computers, the new TI-99/4A will be light grey. Second, the on/off switch has been moved to a more accessible location. (See the photograph below.) With the exception of these two innovations, the new TI-99/4A will remain functionally unchanged from the original model.



The new grey TI-99/4A

In early September, 1983, Dave Loenig, the TI press relations representative, announced that Texas Instruments' present peripheral products would be offered at a reduced, "bundled" price to make the cost of these peripherals more appropriate for an under-$100 computer. (The 99/4A is still selling for $149.95, or less, with a $50 rebate from Texas Instruments.) Through this bundled program, you can obtain $1,200 worth of hardware for $550. The disk drive memory system includes a $250 expansion box, a $250 disk controller, a $400 disk drive, and a $300 memory expansion card with 32K (over 32,000 characters of storage).

As an additional incentive to retail stores, one of three $100 software packages will be given for every bundled package sold. The software choices are word processing, the LOGO 2 educational software, and Microsoft's Multiplan™. Then dealers will be able to offer the selected software at reduced prices—or do as they desire.

This price reduction should boost sales in both 99/4A consoles and the new, more affordable peripherals.

For its computer and calculator products, Texas Instruments provides excellent service and exchange policies, even after warranty periods have expired. Across the country there are over 40 centers at which you can exchange your faulty unit for another at minimal cost. No expensive service contract is needed, as is often the case with many other computer products.

Overall, the TI product line is excellent, and the company's future plans seem bright. Although other vendors may abandon their older machines as they become out of date, Texas Instruments has wisely chosen to make their existing machines more usable for their customers, especially in providing new technology in peripherals for the TI-99/4A.

NOTE: Texas Instruments has a patent pending on the HEX-BUS system and will continue to develop devices for it. The company does not intend to license the patent to others for manufacturing and distributing peripherals for TI computers.

More Computer Knowledge from QUE

TITLE	PRICE
The First Book of (Coleco) Adam	$12.95
The Second Book of (Coleco) Adam: SmartWriter	$ 9.95
TI-99/4A Favorite Programs Explained	$12.95
Timex/Sinclair 1000 Dictionary & Reference Guide	$ 4.95
Timelost for the TI-99/4A	$ 6.95
Timelost for the VIC-20	$ 6.95
Timelost for the Atari	$ 6.95
Timelost for the Timex/Sinclair 1000	$ 5.95
IBM's Personal Computer, 2nd Edition	$15.95
IBM PC Expansion & Software Guide	$16.95
PC DOS User's Guide	$12.95
MS-DOS User's Guide	$12.95
Spreadsheet Software: from VisiCalc to 1-2-3	$15.95
Using 1-2-3	$14.95
1-2-3 for Business	$14.95
Real Managers Use Personal Computers!	$14.95
Multiplan Models for Business	$14.95
SuperCalc SuperModels for Business	$14.95
VisiCalc Models for Business	$14.95
Using KnowledgeMan	$15.95
Using Infostar	$16.95
CP/M Software Finder	$14.95
C Programmer's Library	$19.95
C Programming Guide	$17.95
CP/M Programmer's Encyclopedia	$19.95
Understanding UNIX	$17.95

ORDER FROM QUE TODAY

1-800-428-5331

Timelost

by Kris Andrews, Arlan Andrews, Ph.D., and Joseph C. Giarratano, Ph.D.

Want to enjoy an exciting adventure story and play your own computer games as you read it? This unique book contains a thrilling comic book story and instructions for playing computer games along with the plot. The story follows two kids as they go through a bizarre series of adventures in unknown corners of space and time. You will learn how to make their adventures come alive on your computer's TV screen in several different games, such as "Attack of the Slime Creatures," "Peril of the Pitdemons," and "Rockfall!"

Whether you use a TI-99/4A, VIC-20, Atari, or Timex/Sinclair 1000 home computer, you're sure to enjoy this exciting and challenging adventure comic book. You don't need to know how to write computer programs—the book will teach you everything you need to know to enter and play your own games. You can then use your imagination to change the games as you want.

Use this handy coupon to order your copy of *Timelost* today, or call Que at 1-800-428-5331.

Please send me the following books:

TITLE	PRICE	QUANTITY	TOTAL
Timelost TI-99/4A Version	6.95		
Timelost VIC-20 Version	6.95		
Timelost Atari Version	6.95		
Timelost Timex/Sinclair 1000 Version	5.95		
		Book Subtotal	
		Shipping & Handling ($1.50 per item)	
		Indiana Residents Add 5% Sales Tax	
		GRAND TOTAL	

Method of Payment:

Check_____

Charge My: VISA_____ MasterCard_____ American Express_____

Card Number _____

Expiration Date _____

Cardholder Name _____

Ship To _____

Address _____

_____

City _____ State _____ Zip_____

Mail To:

Que Corporation

7960 Castleway Drive
Indianapolis, IN 46250
(317) 842-7162

TI-99/4A Favorite Programs Explained shows you the surprising power of your TI-99/4A home computer. The tested programs in this book are more than simple examples of BASIC code. They are designed to be truly useful for a variety of practical applications.

Learn how to use your TI-99/4A to analyze savings and investments, calculate basketball team statistics, generate graphs, and much more. Every program has been written to run on the simplest TI-99/4A computer. You don't need extended BASIC, a disk drive, expanded memory, or a printer.

The unique format of this book uses a narrative description alongside each program. Rather than just key in each program, you can read the explanation to follow what is happening in the code and learn more about programming your computer. The final chapter of the book is a handy reference that explains in great detail every TI BASIC command, including illustrative examples to show you how the commands are used.

With this book, you can learn to make the most of your TI-99/4A home computer while gaining skill as a BASIC programmer.

que™

0-88022-050-3