MODULAR

COMPUTER

LESSON

DESIGN

by

Paul M. Roper

and

David V. Loertscher

MODULAR COMPUTER LESSON DESIGN

APPLE VERSION

PRELIMINARY EDITION

by

Paul M. Roper

&

David V. Loertscher

Hi Willow Research and Publishing

1982

Hi Willow Research and Pulishing
Box 1801,
Fayetteville, Arkansas 72702-1801

ISBN: 0-931510-09-0

TABLE OF CONTENTS

INTRODUCTION

Many educators, whether in formal, informal, or corporate education, are becoming computer literate and know the rudiments of programming. These persons know the functions of computer commands like LOAD, LIST, RUN, PRINT...etc. What they may not know is how to use these commands to create a lesson, i.e., they have the tools but do not know how to proceed systematically. The authors have seen a number of beginners try to write lessons. They struggle - not because they don't know how to get the computer to respond, but they get bogged down in hundreds of line numbers and lose their place. They may have used flowcharting techniques but like other programmers desire a better way of structuring their programs.

This book provides a simple structure for a computerized lesson. It breaks a large task or lesson down into a number of pieces, each of which can be programmed or coded separately and then pieced together into a whole. It is something like putting a puzzle or a patchwork quilt together.

Chapter one teaches the technique and provides a detailed example to follow. Chapter two provides a number of tricks that can be put into the lessons to make them more professional.

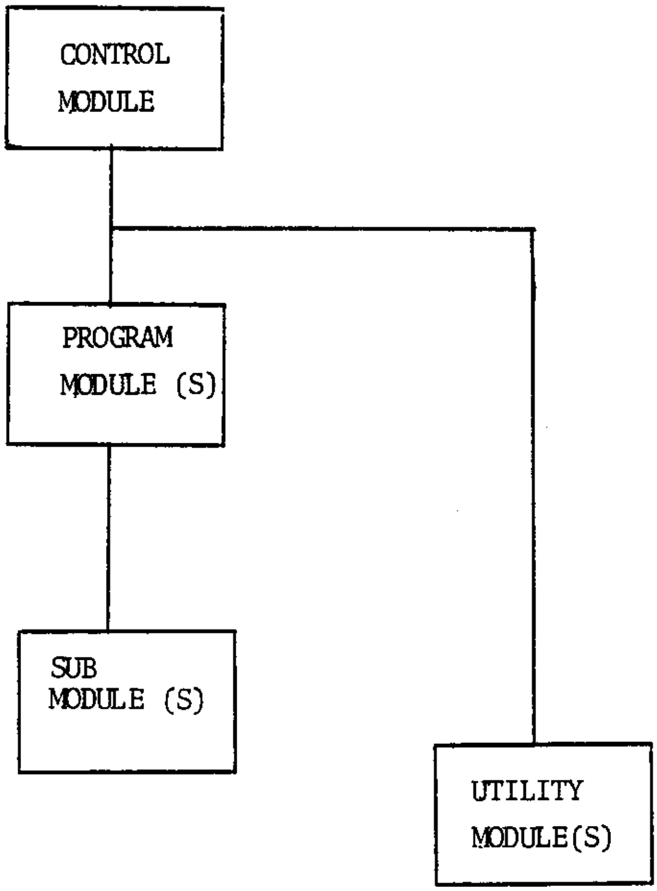
There are a number of commercially available authoring programs on the market such as Genius I, Super Apple Pilot, Blocks...etc. All these have their strong points. They also have limitations. This book provides an alternative to those authoring systems which allows the creative teacher and programmer another way of building lessons for the computer using BASIC. Although the text has been written with Applesoft BASIC in mind, the technique taught here is useful no matter what computer or computer language is employed.

CHAPTER ONE

MODULAR COMPUTER LESSON DESIGN

Modular computer lesson design is a systematic way of creating computerized lessons. Its concept is to divide a programming task down into small segments which can be programmed independently and then pieced together to create an educational lesson. It is similar to the cut and paste technique in the graphic arts where bits and pieces of this and that are combined to create a pleasing handout, poster, etc. Each piece (module) of the computerized lesson can stand alone, is programmed separately, and will run independent of the other modules.

Visually, a computerized lesson would be programmed in the following modules:



The compqnents of each of the modules in the model might include:

1. Control module:
   a. documentation (what the program does)
   b. open any files needed
   c. initialize any variables used
   d. dimension any arrays used
   e. menu
   f. GOSUB statements for the various menu choices
   g. END statement

2. Program modules (are designated as subroutines in the program).
   a. actual lesson content
   b. both text and graphics used only once in the program (graphics or text used over and over should be in the utility module)

3. Sub-modules (are subroutines)
   a. sub-sections of lesson content if it is desirable to break the module down into smaller pieces.

4. Utility modules (are subroutines)
   a. title graphic
   b. graphics called more than once
   c. sound routines called more than once
   d. sorts
   e. time delays
   f. forward or backward paging
   g. keyboard input controls
   h. error trapping

STEP ONE

The first step of modular computer lesson design is to define the lesson problem. Here, the steps of instructional design should be taken into account. This includes an analysis of the intended audience, the objectives of the lesson, the content to be covered, and the strategy that will be employed. The computerized lesson can be independent of other learning materials or it can be one component in a multi-media unit of instruction.

SAMPLE LESSON PROBLEM DEVELOPMENT

Title of sample lesson: Apple Demo

Audience: a student who has mastered the fundamental commands of a programming language and is ready to use those skills to write computerized tutorials.

Objectives:

1. The student will be able to use the sample lesson as a model to follow in the construction of a computerized lesson.
2. The program used in the example will be simple enough that students will be able to follow through the various modules without becoming confused.
3. Enough programming techniques will be demonstrated in the sample lesson so that students can copy, select, add to, and delete ideas as they program their own lessons.
4. A secondary objective is to create a lesson which demonstrates some of the features of the Apple computer.

STEP TWO

The next step in modular design is to break down the main problem into smaller problems. Each of the features of the desired program should be listed. The features are then studied and prioritized according to any constraints that might be present.

SAMPLE LESSON DETAILED FEATURES

Programming features to demonstrate:

1. use of a menu*
2. use of subroutines*
3. documentation within a program*
4. control of input from the keyboard*
5. control of program flow*
6. communication with the user*
7. testing responses from the user*
8. handling errors*

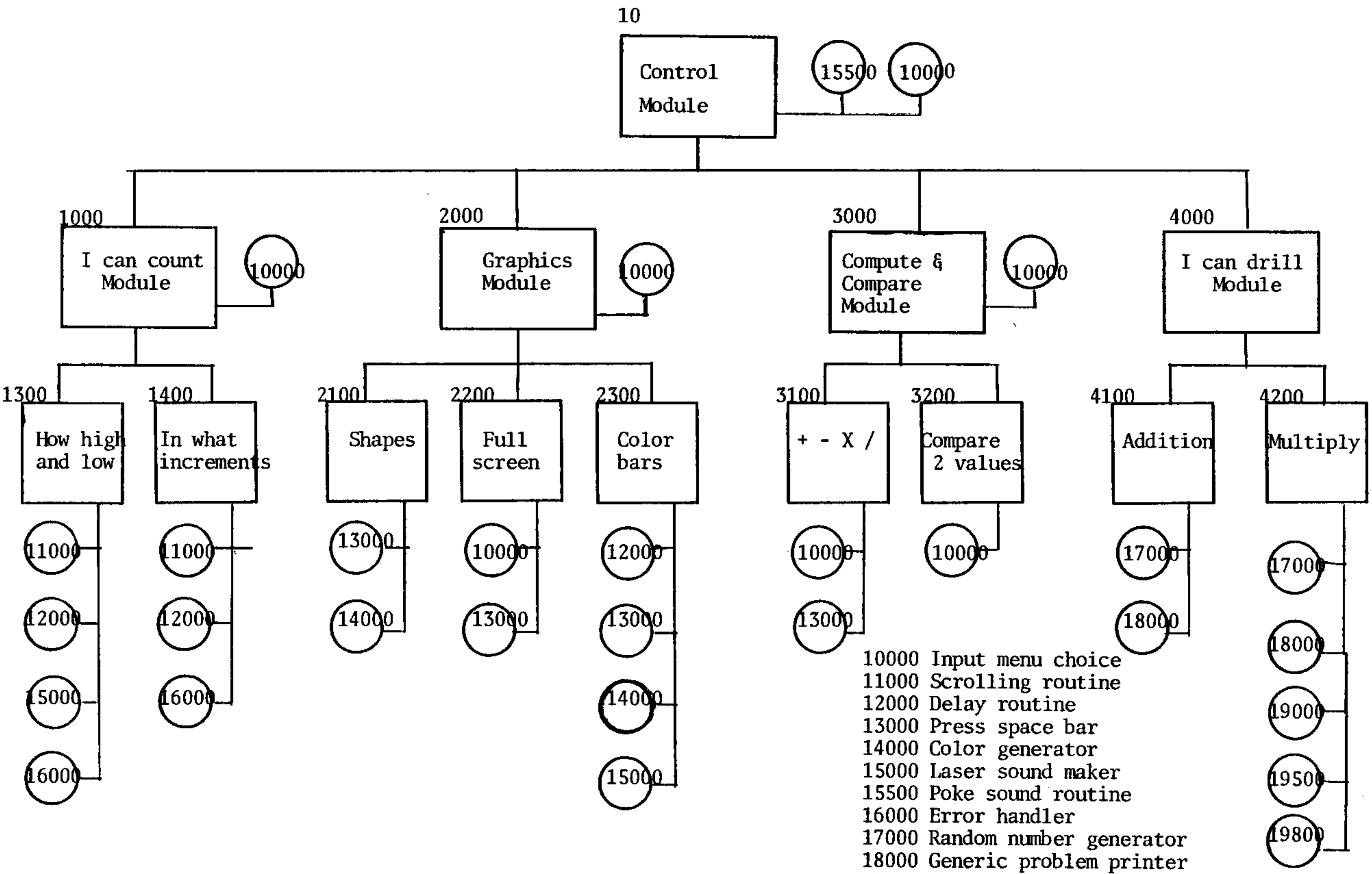
Apple features to demonstrate in the lesson:

1. computers can count*
2. computers can compute and compare*
3. computers can do graphics*
4. computers can create sound
5. tutorial type computer lessons*
6. simulation type computer lessons
7. gaming type computer lessons
8. drill type computer lessons*
9. management of computer lessons

*features chosen for final product

STEP THREE

The next step is to create a VTOC (visual table of contents) of our lesson features. This will be comparable to creating a table of contents for a book which will list module titles (chapter titles) and will give beginning program line numbers of each module (page numbers for the chapters). Line numbers should be added only if they can be easily forecast in advance. Sub-parts of modules (parts of a book chapter) are drawn underneath main modules. Any subroutines referenced by the module are also listed under that module.



10000	Input menu choice
11000	Scrolling routine
12000	Delay routine
13000	Press space bar
14000	Color generator
15000	Laser sound maker
15500	Poke sound routine
16000	Error handler
17000	Random number generator
18000	Generic problem printer
19000	Get math response
19500	Addition problems
19800	Multiplication problems

VTOC (VISUAL TABLE OF CONTENTS)

The VTOC on the previous page lists every subroutine that is called by the program. At the planning stage, the programmer can't forcast every detail of the program so the VTOC may only include the major sections. As thinking and planning progess, the VTOC will become more detailed. At some point, the VTOC will be a complete table of contents to the program like the example above.

STEP FOUR.

The actual programming task is now ready to begin. Each module on the VTOC should be programmed or coded separately. The programmer begins at the top - or the control module and continues in order until the last module is completed. This is called top-down programming.

Each module is an independent piece of the program and can be written and tested before going on to the next module. This can be done by just typing in RUN starting line # of the module, or GOSUB starting line # of the subroutine. It is important to find and clean up problems early in the program rather than having them stack up. Psychologically, this helps create many small successes rather than a mountainous number of problems to tackle all at one time.

One nice thing about programming in modules is that several persons can be assigned various modules of the program to write. Some of the modules may already be be available from other programs that have been written or can be borrowed from other programmers. The idea here is to keep a library of utilities and other useful programs handy that can be pulled into any lesson being written. This could be called a cut and paste methodology - get the modules you need anywhere you can get them and put them together.

The Apple System Master has a very useful utility entitled RENUMBER. Using this program, it is very easy to merge a number of programs together into a single program with one sequence of line numbers. If the two programs to be merged have duplicate line numbers, the RENUMBER program can supply a new sequence of line numbers so that the two programs can merge easily.

If you have a library of usefuly programs or utilities handy, it is very important that each of these have clear titles and descriptions of exactly what they do. A good catalog of what you have is very important!

Be sure to keep the VTOC - even after the program is written. It will always be useful as a key to that program.

In the following pages, the actual program "Apple Demo" is printed out. The program is listed on the left of the page and comments have been added in the right hand column to help the reader follow the logic.

```
10  REM  APPLE DEMO
20  REM  BY MIKE ROPER
30  REM  C. 1982
35  REM ***************
40  REM *CONTROL MODULE*
45  REM *************
50 BEEP$ =  CHR$ (7): REM  STORE BEEP
100  GOSUB 15500: REM  POKE SOUND
110  FOR I = 0 TO 3: READ ADD$(I): NEXT I
120  FOR I = 0 TO 3: READ AANS$(I): NEXT I
130  FOR I = 0 TO 3: READ MULT$(I): NEXT I
135  FOR I = 0 TO 3: READ MANS$(I): NEXT I
325  REM
490  REM  ***************
500  REM  ** MAIN MENU **
505  REM  ***************
510  REM
550  HOME : VTAB 8: HTAB 5
560  PRINT "---- MAIN MENU ----"
570  PRINT : PRINT "   1. I CAN COUNT"
580  PRINT : PRINT "   2. I CAN DO GRAPHICS"
590  PRINT : PRINT "   3. I CAN COMPUTE AND COMPARE"
600  PRINT : PRINT "   4. I CAN DRILL"
610  PRINT : PRINT "   5. END"
630  GOSUB 10000: REM  GET ANSWER
640  ON ANS GOSUB 1000,2000,3000,4000
650  IF ANS <  > 5 THEN 550
660  END
```

The control module begins.

l.110-135 initialize questions and answer variable for the drill module.

main menu is presented.

End statement comes at the end of the control module.

Note that remark statements placed on l. 100 and 620 clarify what will happen at that point in the program.

Note that the menu allows the student to get out of the program. This is a very important characteristic that should be included in almost all computerized lessons.

```
989  REM
990  REM  **********************
1000  REM * I CAN COUNT MODULE *
1005  REM **********************
1008  REM
1010 TIME = 20
1050  REM
1100  HOME : VTAB 10
1200  PRINT "-----   MENU  -----"
1210  PRINT : PRINT
1220  PRINT : PRINT "1. HOW HIGH AND LOW"
1230  PRINT : PRINT "2. IN WHAT INCREMENTS"
1240  PRINT : PRINT "3. RETURN TO MAIN MENU"
1250  GOSUB 10000
1255  IF ANS = 3 THEN  RETURN
1260  ON ANS GOSUB 1300,1400,1500
1270  GOTO 1100
1290  REM  ***************************
1292  REM  *HOW HIGH & LOW SUB MODULE*
1294  REM  ***************************
1300  HOME : VTAB 10
1310  PRINT "I CAN COUNT FROM :": PRINT : PRINT
1315  PRINT " ONE MILLION BELOW ZERO": PRINT
1318  PRINT "TO ONE MILLION ABOVE ZERO": PRINT
1320  PRINT : PRINT "WOULD YOU LIKE ME TO COUNT FOR YOU?"
1321  PRINT : INPUT "(Y OR N)";ANS$
1330  IF  LEFT$ (ANS$,1) = "Y" THEN 1380
1340  HOME : GR : TEXT : GOSUB 15000
1341  FOR I = 1 TO 50: NEXT I
1345  HOME : VTAB 10: HTAB 5
1346  PRINT "OK, BUT I REALLY DID WANT TO...."
1350  GOSUB 11000: HOME : GOTO 1100
1380  HOME : PRINT "SO YOU WON'T BE HERE ALL DAY WAITING"
1381  PRINT : PRINT "FOR ME TO COUNT, KEEP THE NUMBERS SMALL"
1382  PRINT : PRINT "FOR EXAMPLE, FROM 1000 TO 5000"
1383  PRINT : PRINT
1385  INPUT "WHERE SHOULD I START COUNTING ?";BEG
1386  PRINT : PRINT
1388  INPUT "WHERE SHOULD I STOP COUNTING?";QUIT
1390  FOR I = BEG TO QUIT
1392  ONERR  GOTO 16000
1393  GOSUB 12000
1395  PRINT I;" ";
1397  NEXT I
1398  GOSUB 11000: HOME : GOTO 1100
```

Note that a sub-menu has been created and that an exit to the main menu has been provided.

The ON command on line 1260 is a very easy way to skip to various sub-modules.

```
1399  REM  **********************
1400  REM  * INCREMENT S-MODULE *
1405  REM  **********************
1410  HOME : VTAB 5
1415  PRINT "I CAN COUNT BY ANY INCREMENT YOU LIKE.": PRINT
1420  PRINT "I CAN COUNT FORWARD OR BACKWARD"
1430  PRINT : PRINT "(TO MAKE ME PRINT BACKWARD, ENTER": PRINT : PRINT "A
      NEGATIVE NUMBER FOR THE INCREMENT)"
1435  VTAB 15: INPUT "BY WHAT INCREMENT SHOULD I COUNT ?";J: PRINT
1440  PRINT : INPUT "WHERE SHOULD I START COUNTING ?";BEG: PRINT
1450  PRINT : INPUT "WHERE SHOULD I STOP COUNTING ?";QUIT
1465  HOME
1470  FOR I = BEG TO QUIT STEP J
1475  ONERR  GOTO 16000
1480  PRINT I;" ";
1485  GOSUB 12000
1490  NEXT I
1495  GOSUB 11000: GOTO 1100
1500  RETURN
1600  REM
1845  GOSUB 12000: HOME : NEXT J
1990  REM  *******************
2000  REM  * GRAPHICS MODULE *
2005  REM  *******************
2010  REM
2050  TEXT : HOME : VTAB 10: HTAB 10
2060  PRINT "-- I CAN DO GRAPHICS --"
2070  PRINT : PRINT "  1.SHAPES"
2075  PRINT : PRINT "  2.FULL SCREEN ONE COLOR"
2080  PRINT : PRINT "  3.BARS OF DIFFERENT COLOR"
2090  PRINT : PRINT "  4.RETURN TO MAIN MENU"
2092  GOSUB 10000
2093  IF ANS = 4 THEN  RETURN
2094  ON ANS GOSUB 2100,2200,2300
2096  GOTO 2050
```

The VTAB and HTAB commands help position the menus and the text on the screen.

The variables are named so that they are indicative of their function.

Sometimes, REM is used just to create a space in the programming itself to set off sections of the program.

```
2098  REM  *********************
2100  REM  * SHAPES SUB-MODULE *
2101  REM  *********************
2102  FOR N = 1 TO 5
2103  HOME : GR
2105  GOSUB 14000
2107 I = 29:J = 29
2109  FOR K = 4 TO 11
2111  HLIN I,J AT K
2113 I = I - 1:J = J + 1
2115  NEXT K
2118  GOSUB 14000
2120  FOR I = 1 TO 10
2125  HLIN 1,5 AT I
2130  NEXT I
2135  FOR I = 1 TO 10
2140  VLIN 1,5 AT I
2145  NEXT I
2147  GOSUB 14000
2150  FOR I = 13 TO 18
2155  VLIN 0,8 AT I
2160  NEXT I
2165  REM
2170  GOSUB 14000
2175  FOR I = 10 TO 20
2180  HLIN 10,20 AT I
2185  NEXT I
2187  GOSUB 14000
2190  FOR I = 21 TO 23
2195  HLIN 0,39 AT I
2196  NEXT I
2197  GOSUB 13000: NEXT N
2198  RETURN
```

Note that variables are used to set the drawing positions of the horizontal and verticle lines. This cuts down on the number of programming lines that must be written and entered into the computer.

```
2199  REM  *********************
2200  REM * FULL SCREEN COLOR *
2210  REM  *********************
2215  REM
2250  HOME : VTAB 3
2255  PRINT "HERE ARE THE COLORS :": PRINT : PRINT
2260  PRINT "1. MAGENTA","8. BROWN": PRINT
2265  PRINT "2. DARK BLUE","9. ORANGE": PRINT
2270  PRINT "3. PURPLE","10 GRAY": PRINT
2275  PRINT "4. DARK GREEN","11. PINK": PRINT
2280  PRINT "5. GRAY 1","12. LIGHT GREEN": PRINT
2285  PRINT "6. MED. BLUE","13. YELLOW": PRINT
2290  PRINT "7. LIGHT BLUE","14. AQUA": PRINT
2291  PRINT "","15. WHITE"
2292  GOSUB 10000
2293  IF ANS <  = 0 OR ANS > 15 THEN  PRINT BEEP$;: GOTO 2292
2294  HOME : GR : COLOR= ANS
2296  FOR I = 0 TO 39: FOR J = 0 TO 39
2297  PLOT I,J
2298  NEXT J,I
2299  GOSUB 13000: GOTO 2000
2300  REM  **************
2301  REM  * COLOR BARS *
2302  REM  * SUB-MODULE *
2303  REM  **************
2310  REM
2340 TIME = 50
2350  GR : HOME
2355  FOR J = 1 TO 3
2356  GOSUB 15000
2360  FOR I = 0 TO 39
2370  GOSUB 14000: REM  RND COLOR
2380  VLIN 0,39 AT I
2385  NEXT I
2390  GOSUB 12000: NEXT J
2392  GOSUB 13000
2395  RETURN
2400  RETURN
```

The error trapping that is done on line 2293 is a very important part of any program. If the student pushes any other key than that asked for, the computer knows it and responds with some correctional instructions. Sometimes, the question will be repeated. Other times, the student might be reprimanded.

```
2500  REM
2995  REM  *********************
3000  REM  * COMPUTE & COMPARE *
3005  REM  *       MODULE      *
3008  REM  *********************
3009  REM
3010  HOME : VTAB 10
3015  PRINT : PRINT "I CAN COMPUTE AND COMPARE...."
3020  VTAB 14: PRINT "1. ADD,SUB,MULT,DIV"
3025  PRINT : PRINT "2. COMPARE TWO VALUES"
3030  PRINT : PRINT "3. RETURN TO MAIN MENU"
3040  GOSUB 10000
3042  IF ANS = 3 THEN  RETURN
3043  ON ANS GOSUB 3100,3200
3045  GOTO 3010
3048  REM
3050  REM  ********************
3100  REM  * ADD SUB MULT DIV *
3105  REM  *    SUB MODULE    *
3108  REM  ********************
3110  REM
3130  HOME : VTAB 5
3135  PRINT "I CAN PERFORM THE FOLLOWING :"
3138  VTAB 8
3140  PRINT : PRINT "1. ADD NUMBERS"
3142  PRINT : PRINT "2. SUBTRACT NUMBERS"
3145  PRINT : PRINT "3. MULTIPLY NUMBERS"
3147  PRINT : PRINT "4. DIVIDE NUMBERS"
3148  PRINT : PRINT "5. CHOOSE ANOTHER SUBJECT"
3150  GOSUB 10000
3151  IF ANS = 5 THEN  RETURN
3152  IF ANS > 3 THEN 3172
3155  HOME : PRINT "HOW MANY NUMBERS DO YOU WANT TO ENTER"
3158  PRINT : INPUT "(MUST BE 10 OR LESS )";NN
3161  IF NN > 10 THEN 3160
```

ADD SUB MULT DIV SUB MODULE, cont.

```
3162  FOR J = 1 TO NN
3163  PRINT "ENTER NUMBER ";J
3164  INPUT NUM(J): NEXT J
3165  IF ANS = 5 THEN  RETURN
3166  ON ANS GOSUB 3168,3173,3180,3185,3198
3167  GOTO 3130
3168  HOME : VTAB 5: FOR J = 1 TO NN
3169 SUM = SUM + NUM(J): PRINT  SPC( 10);NUM(J)
3170  NEXT J
3171  PRINT  SPC( 8);"+": PRINT  SPC( 8);"--------"
3172  PRINT  SPC( 10);SUM: GOSUB 13000: RETURN
3173  HOME : VTAB 5:SUM = NUM(1): FOR J = 1 TO NN
3174 SUM = SUM - NUM(J + 1): PRINT  SPC( 10);NUM(J)
3175  NEXT J: PRINT  SPC( 8);"-"
3176  PRINT  SPC( 8);"----------": PRINT  SPC( 10);SUM
3179  GOSUB 13000: RETURN
3180 SUM = 1: FOR J = 1 TO NN:SUM = SUM * NUM(J)
3181  PRINT  SPC( 10);NUM(J): NEXT J
3182  PRINT  SPC( 8);"X": PRINT  SPC( 8);"--------"
3183  PRINT  SPC( 10);SUM
3184  GOSUB 13000: RETURN
3185  HOME : INPUT "ENTER NUMBER TO BE DIVIDED";DVND
3186  PRINT : INPUT "ENTER NUMBER TO DIVIDE BY ";DIVSR
3187  PRINT : PRINT : PRINT
3188  PRINT  TAB( 10);DVND;" / ";DIVSR;" = ";DVND
3189  GOSUB 13000: RETURN
```

A considerable amount of effort is made in this part of the program to format the screen exactly the way it is wanted.

The SPC command used above is a little bit better than TAB because SPC moves the printing over the number of spaces indicated right after the last thing printed.

```
3198  REM
3199  REM  **********************
3200  REM  * COMPARE TWO VALUES *
3205  REM  *     SUB MODULE     *
3208  REM  **********************
3210  HOME : VTAB 5
3215  PRINT "I CAN COMPARE:": PRINT
3220  HTAB 5: PRINT "1. NUMBERS"
3225  HTAB 5: PRINT "2. LETTERS"
3230  HTAB 5: PRINT "3. BOTH NUMBERS AND LETTERS"
3235  HTAB 5: PRINT "4.CHOOSE ANOTHER OPTION"
3240  GOSUB 10000
3243  IF ANS = 4 THEN 3010
3245  ON ANS GOSUB 3260,3270,3280
3250  GOTO 3210
3255  REM
3260  GOSUB 3400
3268  RETURN
3270 ITEM$ = "LETTERS"
3275  GOSUB 3300
3278  RETURN
3280 ITEM$ = "NUMBERS AND LETTERS"
3285  GOSUB 3300
3288  RETURN
```

Line 3245 sends the program to a subroutine nearby which in turn calls another subroutine. Subroutines can call other subroutines.

```
3290  REM
3295  REM  ******************
3300  REM  * ALPHABETIC SORT *
3303  REM  ******************
3305  REM
3310  HOME : VTAB 5
3320  PRINT "OK, LET'S COMPARE ";ITEM$
3325  PRINT : PRINT "HOW MANY ";ITEM$
3326  PRINT "DO YOU WANT TO ENTER?"
3327  INPUT "(MUST BE NO MORE THAN 10 ITEMS)";NN
3330  IF NN > 10 THEN 3310
3335  REM  * INPUT THE ITEMS *
3338  REM
3340  FOR I = 1 TO NN
3345  PRINT "ENTER ITEM ";I;: INPUT " ? ";OLD$(I)
3350  NEXT I
3360  HOME : VTAB 8: HTAB 10
3361  PRINT "AS ENTERED     IN ORDER": HTAB 10
3362  PRINT "-------     -------"
3365  FOR I = 1 TO NN: HTAB 12: PRINT OLD$(I): NEXT I
3370 START = 1
3375  FOR I = 1 TO NN
3380  FOR K = START TO NN
3383  IF OLD$(I) <  = OLD$(K) THEN 3387
3385 TEMP$ = OLD$(I):OLD$(I) = OLD$(K):OLD$(K) = TEMP$
3387  NEXT K
3390 START = START + 1
3393  NEXT I
3395  VTAB 10: FOR I = 1 TO NN
3397  HTAB 28: PRINT OLD$(I): NEXT I
3398  GOSUB 13000: RETURN
```

Note that in line 3327, the student is instructed on the limitations of the response. Clear instructions to the student are necessary.

There are many ways to program an alphabetic sort. The sort here is known as a bubble sort.

```
3399  REM
3400  REM  ****************
3405  REM  * NUMERIC SORT *
3410  REM  ****************
3415  REM
3420  HOME : VTAB 5
3425  PRINT "OK, LET'S COMPARE NUMBERS"
3426  PRINT : PRINT "HOW MANY NUMBERS DO YOU WANT TO ENTER ?"
3427  PRINT : INPUT "(MUST BE LESS THAN 10 ITEMS )";NN
3430  IF NN > 10 THEN 3420
3435  FOR I = 1 TO NN: PRINT "ENTER ITEM ";I;: INPUT " ? ";NUM(I)
3438  NEXT I
3440  HOME : VTAB 8: HTAB 10
3441  PRINT "AS ENTERED       IN ORDER": HTAB 10
3442  PRINT "----------       --------"
3445  FOR I = 1 TO NN: HTAB 12: PRINT NUM(I): NEXT I
3460 START = 1
3470  FOR I = 1 TO NN
3475  FOR K = START TO NN
3480  IF NUM(I) <  = NUM(K) THEN 3490
3485 TEMP = NUM(I):NUM(I) = NUM(K):NUM(K) = TEMP
3490  NEXT K
3495 START = START + 1
3500  NEXT I
3505  VTAB 10
3510  FOR I = 1 TO NN
3515  HTAB 28: PRINT NUM(I)
3520  NEXT I
3525  GOSUB 13000: RETURN
```

```
3989  REM
3990  REM  **********************
4000  REM  * I CAN DRILL MODULE *
4005  REM  **********************
4015 TIME = 1000
4020  HOME : VTAB 10
4025  PRINT "I CAN DRILL IN :": PRINT  HTAB 5
4026  PRINT "1. ADDITION": PRINT : HTAB 5
4027  PRINT "2. MULTIPLICATION": PRINT : HTAB 5
4028  PRINT "3. RETURN TO MAIN MENU"
4030  GOSUB 10000
4035  IF ANS = 3 THEN  RETURN
4038  ON ANS GOSUB 4100,4200
4040  GOTO 4020
4045  REM
4090  REM  ******************
4100  REM  * ADDITION DRILL *
4105  REM  ******************
4110  REM
4130  GOSUB 17000: REM  RANDOM NUMBER GENERATOR
4132 PROBLEM$ = ADD$(NUM):ANS$ = AANS$(NUM)
4135  GOSUB 18000: REM  PRINT PROBLEM
4140  RETURN
4190  REM
4195  REM  ******************
4200  REM  * MULTIPLICATION *
4203  REM  ******************
4205  REM
4220  GOSUB 17000
4230 PROBLEM$ = MULT$(NUM):ANS$ = MANS$(NUM)
4240  GOSUB 18000
4250  RETURN
```

Note that the drills above are coded in very few lines. The GOSUB commands here send the program to a generic problem printer and solver. This way, problem set up and solution, plus the formatting on the screen need only be programmed once.

```
9989 REM
9990 REM   *********************
10000 REM  * INPUT MENU CHOICE *
10003 REM  *********************
10020 ONERR  GOTO 16000
10050 VTAB 24: HTAB 1: PRINT "ENTER NUMBER FROM MENU";
10070 GET ANS
10090 RETURN
10092 REM
10095 REM  *********************
11000 REM  * SCROLLING ROUTINE *
11010 REM  *********************
11020 GOSUB 15000
11050 FOR J = 1 TO 25
11060 PRINT
11070 FOR K = 1 TO 100: NEXT K
11080 NEXT J
11090 RETURN
11092 REM
11095 REM  *****************
12000 REM  * DELAY ROUTINE *
12005 REM  *****************
12010 REM
12050 FOR L = 1 TO TIME: NEXT L
12060 RETURN
12065 REM
12070 REM  *******************
13000 REM  * PRESS SPACE BAR *
13005 REM  *******************
13010 REM
13030 VTAB 22
13050 PRINT "PRESS SPACE BAR TO CONTINUE";
13060 GET A$
13070 IF A$ < > CHR$ (32) THEN 13060
13090 RETURN
13095 REM
```

The time delay here is generic. It can be set for different waiting periods based on the needs of the lesson at any given instance.

CHR$ (32) is the ASCII code.

```
13990  REM  *******************
14000  REM  * COLOR GENERATOR *
14005  REM  *******************
14010  REM
14050  COLOR=  INT (14 *  RND (1))    a random color number is generated.
14060  RETURN
14065  REM
14990  REM  *********************
15000  REM  * LASER SOUND MAKER *
15005  REM  *********************
15060  & T255,1
15090  FOR P = 250 TO 50 STEP  - 2
15100  & TP,2
15110  NEXT P
15120  FOR P = 50 TO 250 STEP 2
15130  & TP,2
15140  NEXT P
15150  RETURN
15160  REM ***********************
15490  REM *POKE SOUND ROUTINE*
15495  REM ***********************
15500  FOR I = 768 TO 833: READ P: POKE I,P: NEXT I
15510  DATA  201,84,208,15,32,177,0,32,248,230,138,72,32,183,0,201,44,240
      ,3,76,201,222,32,177,0,32,248,230
15520  DATA  104,134,3,134,1,133,0,170,160,1,132,2,173,48,192,136,208,4,1
      98
15530  DATA  1,240,7,202,208,246,166,0,208,239,165,3,133,1,198,2,208,241,
      96
15540  POKE 1013,76: POKE 1014,0: POKE 1015,3
15550  RETURN
15565  REM
15990  REM  *****************
16000  REM  * ERROR HANDLER *
16005  REM  *****************
16010  REM
16020 E =  PEEK (222)
16050  IF E = 16 OR E = 163 THEN  GOTO 16060
16055  END
16060  POKE 216,0: RESUME
```

```
16989  REM
16990  REM  *****************
17000  REM  * RANDOM NUMBER *
17001  REM  *   GENERATOR   *
17003  REM  *****************
17005  REM
17010 NUM =  INT (3 *  RND (1))
17020  RETURN
17990  REM
17995  REM  *******************
18000  REM  * GENERIC PROBLEM *
18003  REM  *     PRINTER     *
18005  REM  *******************
18006  FOR J = 1 TO 3
18007  HOME : VTAB 5
18008  PRINT "ANSWER THE FOLLOWING BY ENTERING THE"
18009  PRINT : PRINT "LETTER OF THE CORRECT RESPONSE"
18010  VTAB 10
18020  PRINT PROBLEM$
18030  GOSUB 19000: REM  GET ANSWER
18035  IF RES$ = ANS$ THEN 18060
18040  VTAB 18: HTAB 5: PRINT "NO, THAT'S NOT CORRECT..."
18045  GOSUB 12000
18048  NEXT J
18049  VTAB 18: HTAB 1: PRINT  SPC( 40): HTAB 1
18050  PRINT "THE CORRECT ANSWER WAS : ";ANS$
18051 TIME = 2000
18052  GOSUB 12000
18055  RETURN
18060  REM
18065  GOSUB 2300: REM COLOR BARS
18070  TEXT : HOME : PRINT CRES$(NUM)
18075  GOSUB 12000: RETURN
```

```
18990  REM
18995  REM  *********************
19000  REM  * GET MATH RESPONSE *
19003  REM  *********************
19005  VTAB 22
19010  VTAB 24: PRINT "ENTER ";: INVERSE : PRINT "LETTER";: NORMAL : PRINT
     " OF CORRECT RESPONSE";
19020  GET RES$
19030  RETURN
19487  REM
19489  REM  *********************
19490  REM  * ADDITION PROBLEMS *
19495  REM  *********************
19500  DATA  2 + 2 = ?    A. 5     C. 6
                                      B. 2     D. 4,4 + 5 = ?     A. 5
      C. 8
          B. 9    D. 10
19505  DATA  3 + 3 = ?   A. 9    C. 7
                                     B. 0    D. 6,4 + 7 = ?    A. 11   C.
      12
         B. 9    D. 3
19790  DATA  D,B,D,A
19792  REM
19794  REM  *********************
19796  REM  * MULTICA. PROBLEMS *
19798  REM  *********************
19800  DATA  3 X 3 = ?      A. 16    C. 6                           B.  9
         D. 8,4 X 4 = ?     A. 16   C. 15                           B.  8
        D.  9
19805  DATA  2 X 4 = ?      A. 6     C. 12                          B. 8
         D. 10,5 X 2 = ?    A. 8    C. 10                           B. 5
        D. 11
19990  DATA   B,A,B,C
```

Chapter Two

The Bag of Tricks; or, Useful Subroutines

Beginning programmers often see some very useful features of commercial programs that they would like to incorporate into their lesson to make their programs run more smoothly and look more professional. Many of these tricks are not explained very well in some of the common programming reference manuals but are well known to advanced programmers or can be devised by them as needed. The idea for this chapter is to provide a number of common "tricks" that can be used as is or modified to individual lesson needs. Add to this chapter as you see other ideas and figure out how they work. You might also wish to put these short programs on a utility disk and then they are ready to pull in and use at any time for your lesson construction.

Desired feature:

```
      HIT ANY KEY TO CONTINUE
      (getting the student to the next section of the lesson)
```

Necessary command:

```
      10 PRINT "HIT ANY KEY TO CONTINUE"
      20 GET A$                    (any string var. name ok)
```

Comment:
Any key pressed whether on purpose or accidentally will trigger the program to go to the next part of the lesson. That's its disadvantage.

Sample program:

```
      10 HOME
      20 PRINT "WHAT IS 2 + 2?"
      30 VTAB 20
      40 PRINT "PRESS ANY KEY TO SEE THE ANSWER"
      50 GET B$
      60 HOME
      70 PRINT "4 IS THE ANSWER"
```

Desired feature:

PRESS RETURN TO CONTINUE
(getting to student to the next section of the lesson)

Necessary command:

```
10 INPUT "PRESS RETURN TO CONTINUE";A$
```

(A$ can be any string var. name)

Comments:
Using the input command here requires the student to press the return key to continue the lesson. The advantage here is that an accidental hit of any key will not trigger the lesson to advance until the return key is pressed.

Sample program:

```
10 HOME
20 PRINT "WHAT IS 2 + 2?"
30 VTAB  20
40 INPUT "PRESS RETURN TO SEE THE ANSWER"; B$
50 HOME
60 PRINT "4 IS THE ANSWER"
```

Desired feature:

    PRESS SPACE BAR TO CONTINUE
    (getting the student to the next section of the lesson)

Necessary command:

```
10 PRINT "PRESS SPACE BAR TO CONTINUE"
20 GET A$
30 IF A$ = CHR$(32) THEN 50
40 GOTO 20
50 REM REST OF PROGRAM HERE
```

Comment:
CHR$(32) is the Ascii code for the space bar. Any specific character on the keyboard may be used by using the Ascii code for that character (see next page for a list of Ascii codes).

Sample program:

```
10 HOME
20 PRINT "WHAT IS 2+2?"
30 VTAB 20
40 PRINT "PRESS SPACE BAR TO SEE ANSWER"
50 GET Z$
60 IF Z$ = CHR$(32) THEN 80
70 GOTO 50
80 HOME
90 PRINT "4 IS THE ANSWER"
```

ASCII Character Codes

ASCII Code	Display Screen Character	Keystroke	ASCII Code	Display Screen Character	Keystroke
0		Ctrl-@	48	0	Ø
1		Ctrl-A	49	1	1
2		Ctrl-B	50	2	2
3		Ctrl-C	51	3	3
4		Ctrl-D	52	4	4
5		Ctrl-E	53	5	5
6		Ctrl-F	54	6	6
7	(bell)	Ctrl-G	55	7	7
8	(backspace)	Ctrl-H or ←	56	8	8
9		Ctrl-I	57	9	9
10	(linefeed)	Ctrl-J	58	:	:
11		Ctrl-K	59	;	;
12		Ctrl-L	60	<	<
13	(carriage return)	Ctrl-M	61	=	=
14		Ctrl-N	62	>	>
15		Ctrl-O	63	,	,
16		Ctrl-P	64	@	@
17		Ctrl-Q	65	A	A
18		Ctrl-R	66	B	B
19		Ctrl-S	67	C	C
20		Ctrl-T	68	D	D
21	(forward space)	Ctrl-U or →	69	E	E
22		Ctrl-V	70	F	F
23		Ctrl-W	71	G	G
24	(cancel line)	Ctrl-X	72	H	H
25		Ctrl-Y	73	I	I
26		Ctrl-Z	74	J	J
27		Esc	75	K	K
28		n.a.	76	L	L
29		Ctrl-Shift-M	77	M	M
30		Ctrl- ^	78	N	N
31		n.a.	79	O	O
32	space	space bar	80	P	P
33	I	I	81	Q	Q
34	"	"	82	R	R
35	#	#	83	S	S
36	$	$	84	T	T
37	%	%	85	U	U
38	&	&	86	V	V
39	'	'	87	W	W
40	(	(	88	X	X
41	)	)	89	Y	Y
42	*	*	90	Z	Z
43	+	+	91	[	n.a.
44	,	,	92	\	n.a.
45	-	-	93		Shift-M
46	.	.	94		
47	/	/	95		n.a.

n.a. = not available on the Apple II keyboard.

Desired feature:

have a menu and allow student a choice

Necessary command:

```
10 ON A GOTO line#,line#,line#, etc.
```

or

```
10 ON A GOSUB line#,line#,line#,etc.
```

Comment:
The numbers in your menu (for example, 1,2,3) are used by the computer to select the next line to execute. If you type in a 1, the computer goes to the first line number listed after the GOTO or GOSUB. If you type a 2, the computer goes to the second line number listed after the GOTO or GOSUB. Using this command eliminates a series of IF THEN statements such as: IF A=1 THEN GOSUB 1000: IF A=2 THEN GOSUB 2000...etc.

Sample program:

```
  10 HOME
  20 PRINT "WHICH WOULD YOU LIKE?"
  30 PRINT "1. PRINT A LIST"
  40 PRINT "2. PRINT MAILING LABELS"
  50 PRINT "3. ADD TO THE LIST"
  60 PRINT "4. DELETE FROM THE LIST"
  70 PRINT "5. END"
  80 VTAB 20
  90 INPUT "PLEASE ENTER THE NUMBER";A
 100 ON A GOTO 1000,1500,2000,2500,3000
 110 GOTO 80
1000 PRINT "HERE IS YOUR LIST": GOTO 10
1300 PRINT "HERE ARE YOUR MAILING LABELS": GOTO 10
2000 PRINT "YOU CAN NOW ADD TO THE LIST": GOTO 10
2500 PRINT "YOU CAN NOW DELETE FROM THE LIST": GOTO 10
3000 PRINT "THIS IS THE END OF THE PROGRAM"
3100  END
```

Desired feature:
Run one program from within another (where you don't want or can't merge the two)

Necessary commands:

```
10 D$ = CHR$ (4) : REM STORE CTRL-D IN D$
20 PRINT D$; "RUN program name"
```

Comments:
The "program name" above is the name of the program you wish to run from another program. Both programs must be on the same diskette or it could be on another diskette in drive 2 (then say "RUN program name, D2". Remember, if you are going to run one program from another, the first program will be erased from memory and the second one loaded in. This means that to get back to the first program or another one, the second program must also contain the necessary commands listed above.

All values in the first program are lost when the second program is run. For example, if you asked the student's name in the first program, it would not carry over to the second program nor would it be there when you return to the first program. Another disadvantage is that this process will be slow because each program has loading time.

The advantage of running one program from another is that the two programs might be in different languages or could be too big (too many K) to store as a single program.

Desired feature:

Prompt student for response and print error messages for inappropriate response.

Necessary command:

```
     10 IMPUT "ENTER ITEM NUMBER"; P
     20 IF P < 5 and P> 0 THEN 60
     30 MSG$ = "MUST BE 1-5"
     40 GOSUB  1000: REM ERROR MESSAGE PRINTER
     50 GO TO 10
     60 GOSUB 2000: REM CLEAR ERROR MESSAGE LINE
     70 REM PROGRAM SEGMENTS HERE
    999 GO TO 10
   1000 VTAB 20
   1010 PRINT MSG$
   1020 RETURN
   2000 VTAB 20
   2010 PRINT SPC(40)
   2020 RETURN
```

Comments:
The error message must stay on the screen long enough to be read. If you just printed the message and then cleared the screen, the student would hear the beep but would never see the message! Therefore, we add a subroutine (GOSUB 2000) to clear the error message line after a correct response.

MSGTHERE is simply a variable used to see if an error message is on the screen.

SAMPLE:

```
      10 REM USING AN ERROR MESSAGE SUBROUTINE
      15 NOISE$ = CHR$(7) : REM A BEEP STORED IN NOISE$
      20 HOME
      30 VTAB1 : REM KEEP MENU FROM MOVING DOWN SCREEN
      40 PRINT"_____ MENU _____ "
      50 PRINT" (1) GO TO 200"
      60 PRINT" (2) TO TO 300:
      70 PRINT" (3) END"
      80 PRINT:PRINT"ENTER NUMBER OF ACTION DESIRED"
      90 GET C
     100 ON C GO TO 200,300,400
     110 MSG$ = " NUMBER MUST BE 1,2,or3"
     120 GOSUB 1000
     130 GO TO 30
     200 IF MSGTHERE THEN GOSUB 2000
     210 REM PUT PROGRAM SECTION HERE
     220 MSG$ - "MADE IT TO 200"
     230 GOSUB 1000
     240 GO TO 30
     300 IF MSGTHERE THEN GOSUB 2000
     310 REM PUT PROGRAM SEGMENT HERE
```

```
 320 MSG$ = "MADE IT TO 300"
 330 GOSUB 1000
 340 GO TO 30
 400 END
1000 PRINT NOISE$
1010 VTAB 20
1020 PRINT MSG$
1030 MSGTHERE = 1
1040 RETURN
2000 VTAB 20
2010 PRINT SPC(40)
2020 MSGTHERE = 0
2030 RETURN
```

Desired feature:

Use of borders around titles or to highlight text

Program: (written in Integer BASIC) BORDERS

```
 10 CALL -936
 15 S= 16336
 50 GOSUB 400
100 PRINT "THIS IS A PROGRAM TO PRINT"
150 GOSUB 500
250 PRINT "BORDERS AROUND SIGNIFICANT TEXT:
300 GOSUB 500
350 PRINT "THAT YOU MAY WANT TO HIGH LIGHT"
360 GOTO 700
370 REM
375 REM ************************
380 REM ** TAB DOWN AND OVER **
385 REM ************************
388 REM
400 VTAB 15: TAB 5: RETURN
473 REM
475 REM ************************

480 REM ** ADD SPACE & TAB OVER ****

485 REM ******************************

500 PRINT : TAB 5: RETURN
580 REM
600 REM ********************
620 REM ** PRINT STARS ***
630 REM ********************
700 REM
720 VTAB 13
730 TAB 2: PRINT "* * * * * * * * * * * * * * *  * * * * *"
750 SOUND= PEEK (S)- PEEK (S)- PEEK(S)
800 REM
845 REM   *****************************
850 REM   ** PRINT THE RIGHT SIDE ***
855 REM   *****************************
860 FOR J=1 TO 9
870 VTAB 12+J: TAB 40: PRINT "*"
880 FOR I-1 TO 500: NEXT I
885 IF J=9 THEN GOTO 900
890 SOUND= PEEK (S)- PEEK (S)- PEEK(S)
900 NEXT J
1000 REM
1015 REM ****************************
1020 REM ** PRINT THE LEFT SIDE ***
1025 REM ****************************
1050 FOR J=1 to 9
1060 TAB 2
1065 VTAB 12+J: PRINT "*"
1069 FOR I=1 TO 500: NEXT I
1070 SOUND= PEEK (S)- PEEK (S)- PEEK(S)
1080 NEXT J
2000 REM
2005 REM
```

```
2008 REM  ***********************
2010 REM  ** PRINT BOTTOM LINE **
2015 REM  ***********************
2020 REM
2050 VTAB 21: TAB 3: PRINT "* * * * * * * * * * * * * * * * * * *
5000 END
```