Questdata
Volume 2 Issue # 11 ©

SAUCER SQUASHER

by
Mark Bitting

Saucer Squasher is a graphics target game for a Super Elf with at least 1K of memory. You have sixteen shots, displayed in the lower left of the screen, to shoot down a flying saucer five times. Each time a shot is fired, one of the sixteen dots is erased. If the flying saucer is hit, it explodes, and a score dot is placed in the lower right.

To keep the game interesting, each time the flying saucer is hit the postion the shot comes from is moved (and not displayed- good luck!). However, if you are having trouble hitting the saucer from a particular position, the shots keep coming from there until you get it right.

If the flying saucer crosses the screen three times without being hit, it shoots down at your score dots, and erases the top one. If there are no score dots, it places a dot in a minus column. The dots in the minus column must be erased with hits before any dots will be put in the score column.

If there are three dots in the minus column, beware. Another miss and you will lose. If you use up all your shots you will lose. But, if you have four dots in the score column and hit the flying saucer again, you win.

To start the game, press the Input key. The flying saucer will appear at the top left of the screen and travel to the right. Pressing the Input key now fires the shots, which go up from the bar across the bottom of the screen. When the game is over, the program loops back to the beginning and waits for the Input switch to start again.

Here are a few changes which can be made to the program. By changing the code at 0108 through 010E to 89 F6 F6 F6 3A 0F D7 you can continue firing after the sixteen shots are used up, as long as you don't miss. If the game gets too easy for you, change the bytes at 0111 to 2F and 0166 to 4F. Then the flying saucer will only go across twice before shooting at your score. Or to have it go across only once (expert version) change 0111 to 27 and 0166 to 47.

Contrary to the beliefs of my friends and family, the game is honest. If the shot and the flying saucer occupy the same space at the same time, a hit is scored. If they occupy adjacent spaces, though, that's a near miss and doesn't count. So good luck loading the program, and happy saucer-squashing.

Register Usage:

R0	Display area	0300-03FF
R1	Display routine	0011-002E
R2	Display stack	02FC-02FF
R3	Initialize and move saucer	002F-00C9
R4	Main delay	0100-01B4
R5	Hit	0209-029D
R6	Win	01B5-0208
R7	Lose	00CA-00F6
R8	Initial shot position	007E
R9	Shot	
RA, RB, RC, RD	Flying saucer	
RE, RF	General utility	

```
ADDR CODE            COMMENT
0000 F8 00 B1 B3
0004 F8 2F A3
0007 F8 02 B2        Initialization
000A F8 FF A2
000D F8 13 A1 D3
0011 72 70
0013 22 78 22 52
0017 C4 C4 C4
001A F8 03 B0
001D F8 00 A0
0020 80 E2           Video Interrupt Routine
0022 E2 20 A0
0025 E2 20 A0
0028 E2 20 A0
002B 3C 20 30 11
002F E2 69           Turn on TV
0031 F8 03 B9
0034 F8 EF A9
0037 93 59           Screen Blank
0039 89 32 3F
003C 29 30 37
003F F8 F0 A9
0042 F8 FF 59
0045 89 FD FF        Put line across bottom of
                     screen
0048 32 4D
004A 19 30 42
004D 99 BA BB BC BD
0052 93 B7 A4 B8
0056 F8 01 B4 B6
005A F8 02 B5        More Initialization
005D F8 CA A7
0060 F8 B5 A6
0063 F8 09 A5
0066 F8 7E A8
0069 F8 D0 A9
006C F8 55 59 19 59  Put Shots on Screen
0071 F8 E0 A9
0074 F8 55 59 19 59
0079 3F 79 37 7B     Wait for "1" to start game
007D F8 EE A9
0080 F8 20 AA 5A
0084 F8 28 AB
0087 F8 30 AC        Load shot position, put
                     flying saucer on screen
008A F8 38 AD
008D F8 70 5B 5D
0091 F8 F8 5C
0094 D4              Go to delay in R4
```

```
ADDR CODE            COMMENT
0095 0A F6 5A 3B 9D
009A 1A 76 5A
009D 0B 3A A3
00A0 1B 1D 0B
00A3 F6 5B 5D
00A6 3B B0
00A8 1B 1D 0B 76     Move flying saucer
00AC 5B 5D 2B 2D
00B0 0C 3A B5
00B3 1C 0C
00B5 F6 5C
00B7 3B BE
00B9 1C 0C 76 5C 2C
00BE D4              Go to delay in R4
00BF 30 94           Loop back through R4 again
                     before moving saucer
00C1 1D 93 5D        Blank Screen
00C4 2D 8D 3A C2
00C8 30 7D           Go back to load shot & saucer
00CA 97 BE           R7-Lose routine
00CC F8 F1 AE        Set location of dot table
00CF F8 5C A9
00D2 4E 59 09 32 DC
00D7 89 FC 08
00DA 30 D1
00DC F8 40 AF
00DF 7B 2E 8E 3A E0
00E4 7A 2E 8E 3A E5  Razz loser via Q-line
00EA 2F 8F 3A DF
00ED F8 31 A3 D3     Return to beginning & try again
00F1 A6 A9 EF        Dot table
00F4 A9 A9 00
00F7 00 00 00 00 00  (extra space)
00FC 00 00 00 00
0100 99 BF           R4-Delays & scoring
0102 F8 E0 AF
0105 0F 3A 0F        See if player used up all the
                     shots
0108 31 0F D7        If all shots are gone, go to
                     Lose (R7)
010B 00 00 00 00     (extra space for alternate code
                      from text)
010F 8A FD 37 32 64  If saucer is at end, go to
                     "Shoot at score"
0114 31 3A           If shot has been fired skip
                     this delay
0116 F8 02 BF
0119 2F 37 22        Delay with input check
011C 9F 3A 19
011F D3 30 00        Return
0122 7B 99 BF
0125 F8 D1 AF        If shot is fired, turn on Q &
                     erase one shot dot
0128 0F 3A 37
012B 2F 0F 3A 37
012F F8 E1 AF
0132 0F 3A 37
0135 2F 0F
0137 FE FE 5F
013A 89 F6 F6 F6     If shot is at top of screen,
                     turn off Q,
013E 3A 45           reload shot position, go to
                     delay at 0116
0140 08 A9 7A 30 16
0145 09 32 4E
0148 FE 3B 4E D5     Check for hit if it hit, go to
                     "Explosion" (R5)
014C 30 00           If miss, continue at 4E
```

```
ADDR CODE              COMMENT
014E 09 FC 80 59       Add shot
0152 F8 03 BF
0155 2F 9F 3A 55       Delay
0159 09 FF 80 59       Subtract shot
015D 89 FF 08 A9       Move shot up
0161 D3 30 00          Return to R3
0164 7A                Saucer shoots at score dots
0165 F8 57 A9          Turn off Q, set up location
                       of saucer's shot
0168 F8 80 59
016B F8 04 BF
016E 2F 9F 3A 6E       Load shot display, delay, erase
                       shot
0172 93 59
0174 89 FF EF          If shot at bottom go to "Add
                       Minus Dot"
0177 32 9D
0179 89 FC 08 A9       Move shot down
017D 09 32 68          If no score dot, go back to
                       Load shot (0168)
0180 F8 30 AF
0183 8F AE 7B
0186 2E 8E 3A 86       Beep
018A 8F AE 7A
018D 2E 8E 3A 8D
0191 2F 8F 3A 83
0195 93 59             Erase score dot
0197 F8 C1 A3 D3       Go to screen blank at end of R3
019B 30 00             Return
019D 9A AF             Add minus dot
019F 89 FF 09 A9
01A3 09 32 B0
01A6 2F 8F 3A AB D7    If there are 3 minus dots, go to
                       "Lose" (R7)
01AB 89 FF 10 30 A2
01B0 F8 08 59 30 97    Add dot & return
01B5 F8 EF A9          R6-win
01B8 93 59 29          Screen blank
01BB 89 3A B8
01BE F8 F9 AE          Set dot table location
01C1 96 BE
01C3 F8 5B A9
01C6 4E 59 09 32 D4
01CB 19 4E 59 89
01CF FC 07 A9 30 C6    Put notice of success on screen
01D4 89 FF A3 32 DE
01D9 F8 83 A9 30 C6
01DE F8 04 BF
01E1 F8 28 A9
01E4 7B 29 89 3A E5
01E9 F8 28 A9          Delay with tone
01EC 7A 29 89 3A ED
01F1 2F 9F 3A E1
01F5 F8 31 A3 D3       Return to screen blank etc.
                       at 0031
01F9 2B A8 12 A8
01FD 13 B8 00 8B       Dot table
0201 A5 8A B5 AA
0205 AC 53 A5 00
0209 95 BE             R5-Explosion & Scoring
020B F8 7E AE          Set dot table address
020E 1D 93 5D
0211 2D 8D 3A 0F       Blank Saucer
0215 8C FF 18 AC AD
021A F8 04 AA
```

```
ADDR CODE              COMMENT
021D 4E 32 27 5C       Load explosion
0221 8C FC 08 AC
0225 30 1D
0227 F8 04 AB 7B
022B 2F 8F 3A 2B       Sound
022F 7A 2F 8F 3A 30
0234 2B 8B 3A 2A
0238 2A 8A 32 42
023C 80 FF 08 AC       If last frame of explosion,
                       blank screen & continue
0240 30 1D             If not, go back to Load at
                       021D
0242 93 5C
0244 2C 8C 3A 42
0248 F8 C6 A9
024B 09 3A 65          Check for minus dots.  If there
                       are any, go to blank.
024E 89 FC 10 A9
0252 89 FF F6
0255 3A 4B 19
0258 09 32 69
025B 89 FF 10 A9       Find location where next dot
                       belongs and go to Add dot
025F 89 FF A7          If this is the fifth hit, go
                       to "Win" (R6)
0262 3A 58 D6
0265 93 59 30 6C       Blank minus dot & go to Move
                       shot location
0269 F8 80 59          Add score dot
026C 08 FF 01 58
0270 08 FF EA 3A 78
0275 F8 EE 58          Move shot location
0278 F8 7D A3
027B D3 30 09          Return
027E 0A 28 1C 0A       Dot table
0282 28 00 09 9C
0286 3E FF 7C 39
028A 90 00 09 9C
028E 3E FF 7C 39
0292 90 00 10 44
0296 11 84 22 08
029A 22 08 00 00       The End!
```

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

Publisher .....Quest Electronics
Editor .....Paul Messinger
Proof Reading .....Judy Pitkin
Production .....John Larimer

E~BUG

by
Phillip Lieschesk

This monitor, which I call EBUG, allows you to load a program and inspect code via an ASCII keyboard and a video board or printer. The following is a brief description of the routines used.

EBUG: Elf Bug Monitor.
This is the main driver routine. It initializes the stack pointer R2, standard subroutine call pointer R4, standard subroutine return pointer R5, break-point interrupt register RE and break-point stack pointer RD. It outputs a carriage return, linefeed and a prompt which is a greater-than sign (>). This prompt signifies an operation code request. The five legal input code are L-load, I-inspect, E-execute, D-dump and B-break-point. If an illegal character is entered, it will not be accepted and a new prompt will be issued.

LOAD: Load
This operation allows the operator to sequentially modify the contents of memory. After entering L, the starting address for loading is entered followed by a carriage return. A 4-digit hexadecimal number must be entered with a CR following. Without a CR, this operation will be aborted. After the CR, the memory address is displayed with a colon following. After the colon, a 2-digit hexadecimal number is entered and followed with a CR. This will allow the operator to modify the next location. Blanks can be used without discretion. To abort or terminate this operation, one merely enters another character other than the CR when the CR is expected.

INSP: Inspection
This operation allows the operator to inspect a single memory location at one time. After the I, the 4-digit memory address is entered and followed by the CR. With this, the address is displayed followed by a colon and then the contents of that memory location. If a CR is entered, the next memory location is displayed. If another character is entered, this operation is aborted.

EXEC: Execute
This operation allows the operator to exit the monitor and enter some other program. After the E, the 4-digit starting address is entered followed by a CR. The program starting at the entered address is executed using R3 as the program counter.

DUMP: Dump
This operation allows the operator to inspect 8-bytes of memory at one time. After the D, the 4-digit starting memory address is entered followed by the CR. With this, the address is displayed with its colon and eight bytes of memory are displayed on that line. Entering a CR allows the operator to inspect the next 8 bytes of memory. Entering another code aborts this operation.

BRKP: Break-Point
This operation allows for simple one-level nested break-point executions of a program. After the B, the 4-digit starting address is entered and then the 4-digit break-point address is entered followed with a CR. A blank can be used to separate these two addresses. After entering the CR, the code is executed until the break-point is reached. With this the contents of the accumulator D is displayed and EBUG is re-entered. It should be noted that during BP execution, RD and RE must not be tampered with by the program. Basically, the break-point operation requests the starting address and pushes it on main stack. Then the break-point address is requested. With this the contents of the break-point location is saved on the break-point stack followed by the break-point address. After this, the SEP RE instruction is stored at the b - p location and the program is executed. Control is returned back to the monitor via the SEP RE. After return, R3 is made the PC; the contents of D is displayed and the original contents of b - p location is restored with help from the b - p stack. The monitor is then re-entered and ready for the next operation. A program entered via this monitor can use the stack pointer R2 without need for initializing and can use the SCRT. Finally the program is able to call and use the subroutines contained in the monitor via the SCRT, but they must use R3 as PC and R2 as stack pointer.

Brief Description of Subroutines:

OTT- Out Teletype
This routine is supplied by the user. It is not part of the monitor package. This allows for output device flexibility. OTT is required to print or display the ASCII character contained in the accumulator D. It returns control back to the calling routine with a SEP R5 instruction. Four bytes starting at location 0E00 have been allotted for a patch into OTT.

IKB- In Keyboard
This routine is similar to OTT. It is required to accept an ASCII character for some input device and loads its code into accumulator D. It returns by a SEP R5 instruction and three bytes starting a 0E04 are reserved for a long branch into IKB. Finally, IKB must echo its characters.

IADR- Input Address
This routine accepts a 4-digit ASCII coded hexadecimal number and converts it into a 16-bit binary number which is stored in R0. It calls IHEX.

OADR- Output Address
This routine takes the 16-bit binary number contained in R0 and converts it into a 4-digit ASCII coded hexadecimal number which is displayed. It also pushes a colon after the displayed number in order to signify that it is a memory address. It calls OHEX and OTT.

INCR- Input a Carriage Return
This routine checks for a carriage return. If no CR is received, the current operation is aborted and the monitor is initialized and re-entered. If a CR is received, a linefeed (LF) is echoed and the current operation is continued. It calls IKB and OTT.

IHEX- Input a Hexadecimal Number.
This routine accepts a 2-digit ASCII coded number and converts it into an 8-bit binary number which is loaded into the accumulator D. It calls IHXD.

OHEX- Output a Hexadecimal Number
This routine accepts an 8-bit binary number contained in the accumulator D and converts it into a 2-digit ASCII coded number which is displayed on the output device. It calls OTT.

IHXD- Input a Hexadecimal Digit
This routine accepts an ASCII coded Hexadecimal digit from the input device and converts it into a 4-bit binary number which is loaded into the least significant nibble (LSN) of D. The most significant nibble (MSN) contains zero. The routine accepts blanks but does nothing with them. This allows for number spacing. It checks to see if the entered character is a legal hexadecimal digit (ie. 0-9, A-F). If the code is illegal, an underline character is presented and that character is not accepted. This routine has no provisions for corrections. It calls IKB.

Summary of EBUG use:

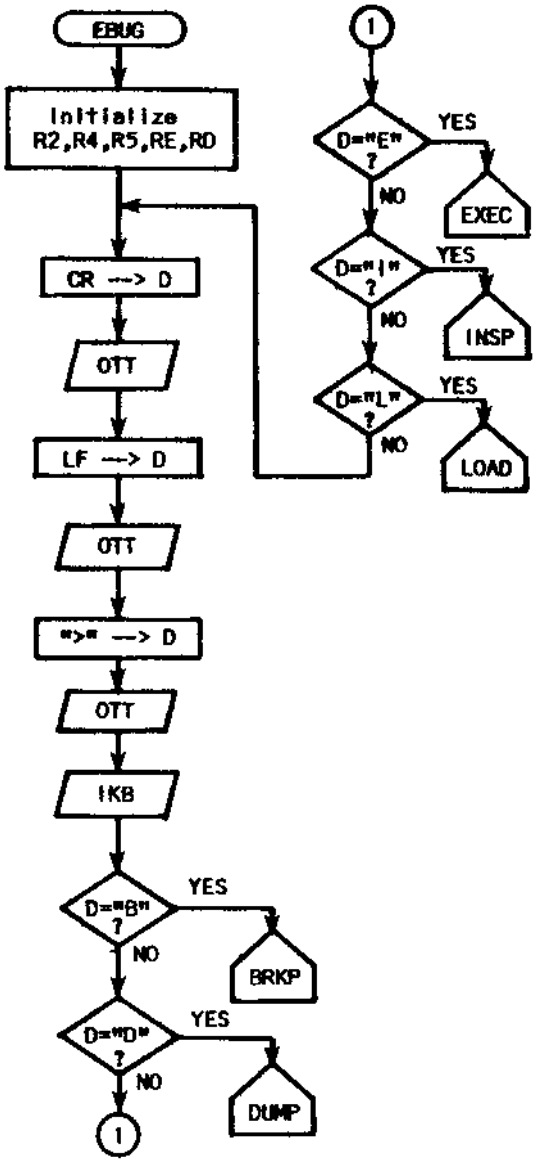
Execute:
>E starting address (CR)

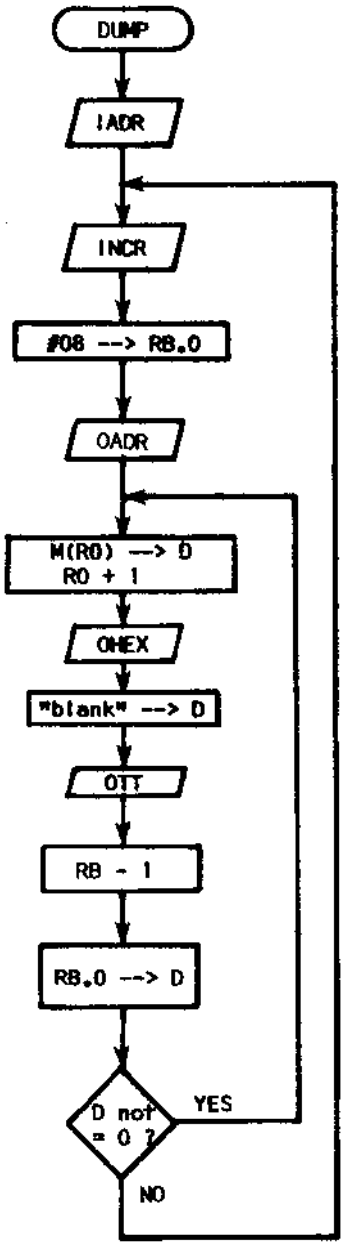
Inspect:
>I starting address (CR)
XXXX: XX (CR)

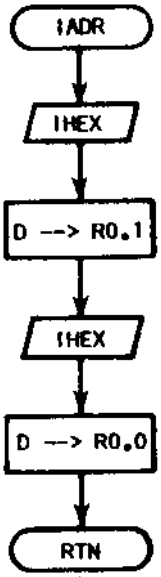
Load:
>L starting address (CR)
XXXX: XX (CR)

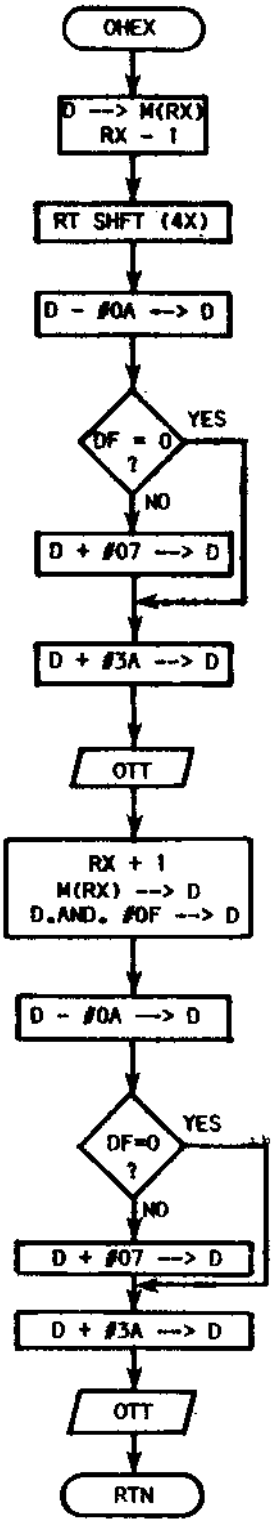
Dump:
>D starting address (CR)
XXXX: XX XX XX ... XX (CR)

Break-Point:
>B starting address ending address (CR)

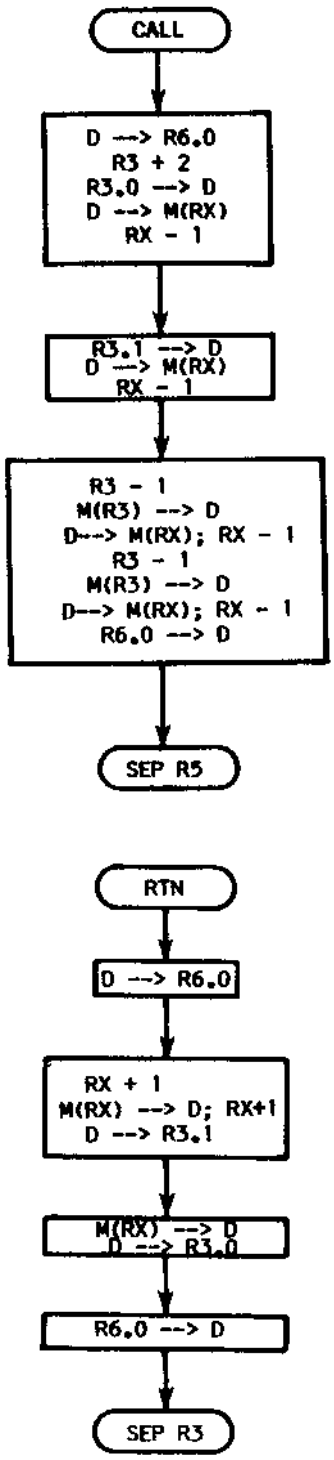


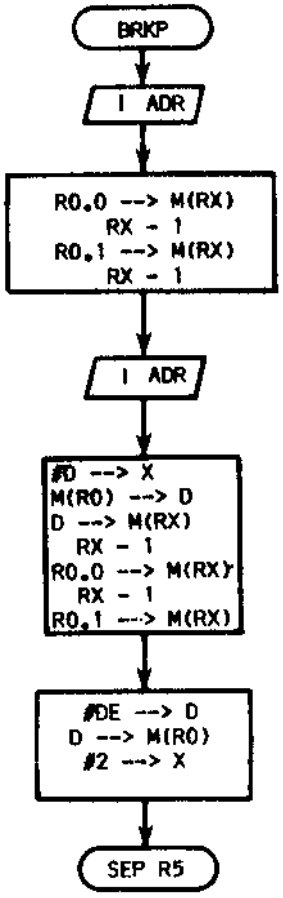


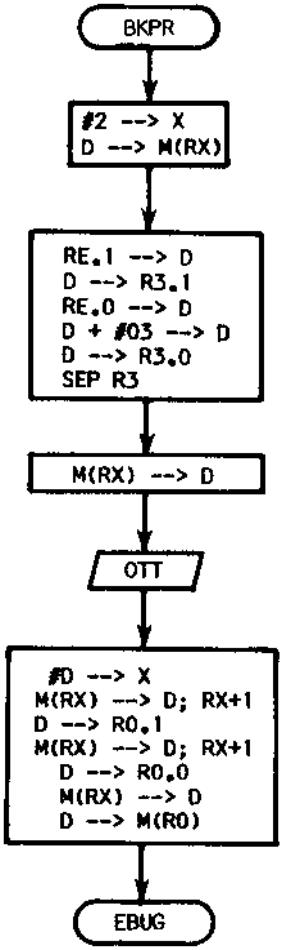


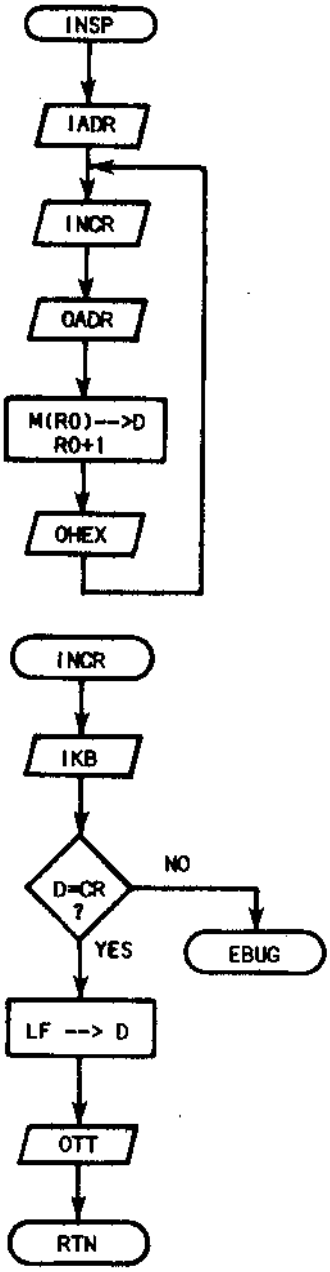


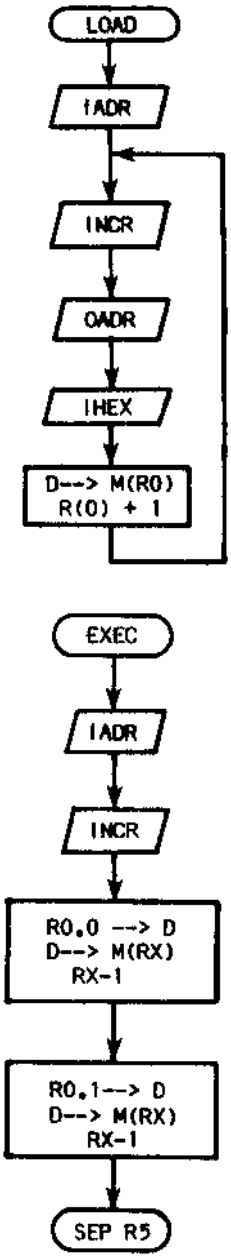
SCRT Routines











Register Used:

R0 = Address Pointer
R2 = Stack Pointer
R3 = Main PC
R4 = Call Routine Pointer
R5 = Return Routine Pointer
R6.0 = SCRT Scratch Pad
RB = Scratch Pad
RD = Break point Stack Pointer
RE = Break point Interrupt Pointer

ADDR	CODE	COMMENT
0E00	BF	
0E01	C0 83 03	OTT Patch
0E04	C0 83 00	IKB Patch
0E07	C0 XX XX	Break-Point Re-entry(Future)
0E0A	E2	EBUG Entry
0E0B	F8 FF	Initialize Registers
0E0D	A2 AD	
0E0F	F8 0E	

```
ADDR CODE            COMMENT
0E11 B4 B5 BE
0E14 F8 98
0E16 B2
0E17 F8 0F
0E19 BD
0E1A F8 4B
0E1C A4
0E1D F8 5D
0E1F A5
0E20 F8 07
0E22 AE
0E23 F8 0D           Give a Prompt
0E25 D4 0E 00
0E28 F8 0A
0E2A D4 0E 00
0E2D F8 3E
0E2F D4 0E 00
0E32 D4 0E 04        Get Op Code
0E35 FF 42           Test if "B" (illegal)
0E37 32 23
0E39 FF 02           Test if "D" (illegal)
0E3B 32 23
0E3D FF 01           Test if "E"
0E3F 32 87
0E41 FF 04           Test if "I"
0E43 32 78
0E45 FF 03           Test if "L"
0E47 32 68
0E49 30 23           Illegal Code
     * SCRT Call
0E4B E2              Set R2 as Stack Pointer
0E4C A6              Save D
0E4D 13 13           Push Returning Address
0E4F 83              on stack
0E50 73
0E51 93
0E52 73
0E53 23
0E54 03              Push Calling address
0E55 73              on stack
0E56 23
0E57 03
0E58 73
0E59 86              Restore D
0E5A D5              Jump to Return
0E5B 30 4B
     * SCRT Return
0E5D E2              Set R2 as Stack Pointer
0E5E A6              Save D
0E5F 60              Pop address on stack
0E60 72              and load into R3
0E61 B3
0E62 F0
0E63 A3
0E64 86              Restore D
0E65 D3              Set R3 as PC
0E66 30 5D
     * LOAD
0E68 D4 0E EF        Call IADR
0E6B D4 0E 92        Call INCR
0E6E D4 0E F8        Call OADR
0E71 D4 0E C3        Call IHEX
0E74 50              Store the Number
0E75 10              Bump Memory Pointer
0E76 30 6B           Do it again
     * INSP
0E78 D4 0E EF        Call IADR
0E7B D4 0E 92        Call INCR
0E7E D4 0E F8        Call OADR
0E81 40              Inspect Memory Location
0E82 D4 0E D1        Call OHEX
0E85 30 7B           Do it again
```

```
ADDR CODE            COMMENT
     * EXEC
0E87 D4 0E EF        Call IADR
0E8A D4 0E 92        Call INCR
0E8D 80              Push Starting Address
0E8E 73              on stack
0E8F 90
0E90 73
0E91 D5              Go to it
     * INCR
0E92 D4 0E 04        Call IKB
0E95 FF 0D           Check for CR
0E97 3A 0A           If not, re-enter EBUG
0E9A F8 0A           If so, echo a LF
0E9C D4 0E 00
0E9F D5              Return
     * IHXD
0E9F D4 0E 04        Call IKB
0EA2 FF 20           Check if a blank
0EA4 32 9F           If so, ok
0EA6 FF 10           Check if legal
0EA8 3B BC           If not, yell
0EAA FF 0A           Check if 0-9
0EAC 33 B1           If so, then add #0A
0EAE FC 0A           and return
0EB0 D5
0EB1 FF 07           Check if legal
0EB3 3B BC           If not, yell
0EB5 FF 06           Check if legal
0EB7 33 BC           If not, yell
0EB9 FC 10           Must be A-F, so add #10
0EBB D5              and return
0EBC F8 3F           Since it is illegal, display
0EBE D4 0E 00        a "?" and try again.
0EC1 30 9F
     * IHEX
0EC3 D4 0E 9F        Get MSN and shift
0EC6 FE FE FE FE     it up in D
0ECA 73              Push on Stack
0ECB D4 0E 9F        Get LSN
0ECE 60              Add with MSN on Stack
0ECF F4
0ED0 D5              Return
     * OHEX
0ED1 73              Save D on Stack
0ED2 F6 F6 F6 F6     Position MSN in D
0ED6 FF 0A           Convert to ASCII code
0ED8 C7
0ED9 FC 07
0EDB FC 3A
0EDD D4 0E 00        Call OTT
0EE0 60              Restore D from Stack
0EE1 F0
0EE2 FA 0F           Mask LSN in D
0EE4 FF 0A           Convert to ASCII code
0EE6 C7
0EE7 FC 07
0EE9 FC 3A
0EEB D4 0E 00        Call OTT
0EEE D5              Return
     * IADR
0EEF D4 0E C3        Call IHEX
0EF2 B0              D --> R(0).1
0EF3 D4 0E C3        Call IHEX
0EF6 A0              D --> R(0).0
0EF7 D5              Return
     * OADR
0EF8 90              R(0).1 --> D
0EF9 D4 0E D1        Call OHEX
0EFC 80              R(0).0 --> D
0EFD D4 0E D1        Call OHEX
0F00 F8 3A           Display a ":"
0F02 D4 0E 00
0F05 F8 20           and then a "blank"
0F07 D4 0E 00
0F1A D5              Return
```

MEMBERSHIP

by
E.H. Sandelin

This program provides the data needed by a club's officer's to maintain the membership rolls. It will also provide mailing labels. The program is written in Quest Super Basic V3.0. It includes a Sort Routine that can sort the membership file five different ways. I wrote the program for a radio control aircraft club, but it can be modified to fit the requirements of any type of club or group. Record fields can be added or removed to tailor the program to your needs. The program is in modular form with most or all of a function within a module. The program logic should be easy to follow from the listing. The sort is a "Schell-Metzner" sort. It's not one of the fastest sorts, but works well. A quick look at the listing for the sort will show how small a routine it is. To prevent an error 54 a sort should always be performed before any list function. The functions provided are add, edit, list (member list or mailing labels), initialize, save, load, sort and delete. The use of arrays allows ease of handling of numeric data and simplifies the sorts. Dates are entered 'YYMM' so they can be sorted. The day of the month is not used, but may be added if needed. Renumbering of the basic statements is not recommended as the GOSUB statement numbers are calculated by the program. See line number 200.

FUNCTION DESCRIPTIONS ;

INITIALIZE	- Clears arrays and sets up for initial data entry
ADD	- Adds new records
DELETE	- Removes entries from file
EDIT	- Edits records in file
SORT	- Sorts records in file
LOAD	- Loads data from tape
SAVE	- Saves program and data on tape
LIST	- Prints member list or mailing labels
END	- Ends program. Save must be done before end

To change the record field descriptions change the following statements ; 1070, 1080, 1090, 1100, 1120, 1126, 1130, 1136, 1140, 1150 and 1156. To change the list heading, statements 3030 thru 3038 must be changed.

```
10 REM  MEMBERSHIP PROGRAM
20 REM       BY E.H. SANDELIN  AUG. 1980
25 REM            WRITTEN IN QUEST SUPER BASIC V3.0
30 V=0
40 DIM V(5): FOR N=1 TO 5:V(N)=0: NEXT N
70 IF V<>1234 GOTO 100
80 PRINT
90 PRINT "FILE SIZE " ;V(3);"  RECORDS AVAILABLE "
        V(3)-V(2)
100 PRINT : PRINT "SELECT AN OPTION ;"
110 PRINT "1 - ADD, 2 - EDIT, 3 - LIST, 4 - DELETE,
         5 - SAVE"
120 INPUT "6 - LOAD, 7 - SORT, 8 - INITIALIZE,
         9 - END   " O
170 IF O<1 GOTO 190
175 IF O>9 GOTO 190
180 GOTO 200
190 PRINT : PRINT "INVALID SELECTION" : GOTO 70
200 GOSUB O*1000+10
300 PRINT : GOTO 70
400 PRINT : PRINT "*****  REQUEST EXCEDES FILE SIZE
         OF " ;V(3): PRINT
405 RETURN
1010 V(1)=1000: PRINT "ADD RECORDS"
1014 PRINT : PRINT "ENTER A RECORD # OF 0 TO END
          ADD" : PRINT
1020 INPUT "RECORD #  " N
1030 IF N>V(3) GOSUB 400: GOTO 1020
1040 IF N<=0 RETURN
1050 IF N(N)=N PRINT "THAT NUMBER ASSIGNED TO "
            ;N$(N): GOTO 1020
1060 N(N)=N
1070 INPUT "MEMBERS NAME  " T$
1075 IF MID$(T$,1,1)="." GOTO 1080
1076 N$(N)=T$
1080 INPUT "STREET ADDRESS  " T$
1082 IF MID$(T$,1,1)="."  GOTO 1090
1084 A$(N)=T$
1090 INPUT "CITY  " T$
1092 IF MID$(T$,1,1)="." GOTO 1100
1094 C$(N)=T$
1100 INPUT "STATE ZIP  " T$
1102 IF MID$(T$,1,1)="." GOTO 1120
1104 S$(N)=T$
1120 INPUT "TELEPHONE    ACD XXX XXXX" T$
1122 IF MID$(T$,1,1)="."  GOTO 1126
1124 T$(N)=T$
1126 INPUT "AMA NUMBER  " A
1127 IF A=0 GOTO 1130
1128 A(N)=A
1130 INPUT "DATE JOINED CLUB   YYMM" J
1132 IF J=0 GOTO 1136
1134 J(N)=J
1136 INPUT "DATE STARTED MODELING   YYMM" M
1137 IF M=0 GOTO 1140
1138 M(N)=M
1140 INPUT "DATE OF BIRTH   YYMM" B
1142 IF B=0 GOTO 1150
1144 B(N)=B
1150 INPUT "RATING  " T$
1152 IF MID$(T$,1,1)="."  GOTO 1156
1154 R$(N)=T$
1156 INPUT "OFFICE " T$
1157 IF MID$(T$,1,1)="."  GOTO 1160
1158 O$(N)=T$
1160 IF V(1)<>1000 RETURN
1190 V(2)=V(2)+1: PRINT : GOTO 1020
```

```
2010 PRINT : PRINT "EDIT RECORDS" : PRINT
2015 V(1)=2000
2017 PRINT : PRINT "ENTER A '.' TO DUP AN ENTRY."
          : PRINT
2020 INPUT "RECORD #  " N
2030 IF N>V(3) GOSUB 400: GOTO 2020
2032 IF N=0 RETURN
2035 IF N(N)<>N PRINT N;"  HAS NOT BEEN ASSIGNED YET"
          : PRINT : GOTO 2020
2050 PRINT N$(N);
2052 PRINT TAB(21);"TEL " ;T$(N);
2054 PRINT TAB(40);"AMA " ;INUM(A(N));
2056 PRINT TAB(54);R$(N);
2058 PRINT TAB(77);INUM(N(N))
2060 PRINT A$(N);
2062 PRINT TAB(36);"DOB " ;INUM(B(N));
2064 PRINT TAB(50);"JOINED " ;INUM(J(N));
2066 PRINT TAB(65);"ST.MDL. " ;INUM(B(N))
2068 PRINT C$(N);", " ;S$(N);
2070 PRINT TAB(40);"OFFICE " ;O$(N)
2075 IF V(1)=3000 RETURN
2078 PRINT : GOSUB 1070
2080 PRINT : GOTO 2020
3010 PRINT : INPUT "1 - LIST, 2 - LABELS  " B
3012 PRINT : INPUT "# OF COPIES  " C
3014 IF B=2 GOTO 3300
3015 V(1)=3000: PRINT : INPUT "TODAYS DATE
          MM/DD/YY" D$
3020 INPUT "MAKE PRINTER READY  " Z$
3030 DATA "REPLACE WITH CLUB NAME"
3032 DATA "STREET ADDRESS"
3034 DATA "CITY, STATE AND ZIP CODE"
3036 DATA "MEMBERSHIP LIST - ACTIVE MEMBERS ONLY"
3038 DATA "ADDITIONAL HEADING DATA" ,"MORE
       HEADING DATA"
3040 FOR A=1 TO 6: READ H$(A): NEXT A
3045 RESTORE
3050 PRINT TAB(25);H$(1);TAB(70);D$
3051 PRINT TAB(25);H$(2)
3052 PRINT H$(5);TAB(25);H$(3);TAB(62);H$(6)
3060 PRINT
3070 PRINT "----------------------" ;H$(4);
       "----------------------" : PRINT
3080 P=14
3100 FOR I=1 TO V(3)
3110 IF S(I,1)=0 GOTO 3150
3115 N=S(I,1)
3120 GOSUB 2050
3130 PRINT
3135 P=P-1: IF P=0 CLS:P=15; PRINT : PRINT
3150 NEXT I
3200 C=C-1: IF C=0 CLS: RETURN
3210 CLS
3220 GOTO 3050
3300 PRINT : INPUT "READY PRINTER  " Z$
3310 FOR I=1 TO V(3)
3320 IF S(I,1)=0 GOTO 3340
3325 N=S(I,1)
3330 PRINT N$(N);TAB(30);INUM(A(N))
3332 PRINT A$(N)
3334 PRINT C$(N)
3336 PRINT S$(N)
3338 PRINT : PRINT
3340 NEXT I
3350 C=C-1: IF C=0 CLS: RETURN
3360 GOTO 3310
4010 PRINT : PRINT "DELETE RECORDS" : PRINT
4015 V(1)=4000
4020 INPUT "RECORD #  " N
4030 IF N>V(3) GOSUB 400: GOTO 4020
4040 IF N=0 RETURN
4050 IF N(N)=0 PRINT N;" HAS NOT BEEN ASSIGNED"
          : GOTO 4020
4052 PRINT : PRINT "RECORD # " ;INUM(N);" IS "
       ;N$(N)
4054 INPUT "VERIFY DELETE   Y/N  " Z$
4056 IF MID$(Z$,1,1)="Y"  GOTO 4059
4058 GOTO 4020
4059 B(N)=0:J(N)=0:M(N)=0:A(N)=0
4060 N(N)=0: PRINT N$(N);"  HAS BEEN DELETED
4062 V(2)=V(2)-1
4070 PRINT : GOTO 4020
5010 PRINT : PRINT
5015 V(1)=5000
5020 INPUT "MAKE TAPE UNIT 2 READY TO RECORD  " Z$
5040 PSAVE : DSAVE
5050 PSAVE : DSAVE
5060 PRINT "DATA HAS BEEN SAVED "
5070 RETURN
6010 PRINT : INPUT "MAKE TAPE READY  " Z$
6015 V=1234
6030 DLOAD
6050 RETURN
7010 PRINT "SORT ROUTINE"
7012 V(1)=7000
7014 GOSUB 7300
7016 IF S=5 GOTO 7230
7020 P=0:M=V(3)
7030 M=INT(M/2)
7040 IF M=0 GOTO 7230
7050 P=P+1: PRINT "PASS " ;P;" STARTED"
7070 FOR L=1 TO M
7080 I=L:J=L+M:W=0
7110 IF Z=1 GOTO 7118
7114 IF S<I,2>>=S(J,2) GOTO 7160
7116 GOTO 7120
7118 IF S(I,2)<=S(J,2) GOTO 7160
7120 W=1
7130 X=S(I,2):Y=S(I,1)
7140 S(I,2)=S(J,2):S(I,1)=S(J,1)
7150 S(J,2)=X:S(J,1)=Y
7160 I=J:J=J+M
7180 IF J<V(3) GOTO 7110
7190 IF W=0 GOTO 7210
7200 GOTO 7080
7210 NEXT L
7220 GOTO 7030
7230 PRINT : PRINT "SORT DONE"
7240 RETURN
7300 PRINT : PRINT "SELECT SORT FIELD ;"
7310 PRINT "1 - JOINED, 2 - DOB, 3 - ST.MDL,
          4 - AMA #, 5 - RECORD #  " ;
7320 INPUT S
7332 IF S<1 GOTO 7300
7334 IF S>5 GOTO 7300
7340 PRINT : INPUT "1 TO SORT ASCENDING, 2 FOR
          DESCENDING " Z
7342 PRINT
7344 IF Z>2 GOTO 7340
7345 IF S=1 FOR I=1 TO V(3):S(I,1)=N(I):S(I,2)=J(I):
       NEXT I
7350 IF S=2 FOR I=1 TO V(3):S(I,1)=N(I):S(I,2)=B(I):
       NEXT I
7355 IF S=3 FOR I=1 TO V(3):S(I,1)=N(I):S(I,2)=M(I):
       NEXT I
7360 IF S=4 FOR I=1 TO V(3):S(I,1)=N(I):S(I,2)=A(I):
       NEXT I
7365 IF S=5 FOR I=1 TO V(3):S(I,1)=N(I):S(I,2)=N(I):
       NEXT I
7370 RETURN
8010 PRINT : PRINT "INITIALIZING"
8015 CLD: DIM V(5)
8020 FOR N=1 TO 5:V(N)=0: NEXT N
8022 PRINT : INPUT "ENTER MAX FILE SIZE, # OF
          RECORDS. " V(3)
8024 DIM A(V(3)),B(V(3)),J(V(3)),M(V(3)),N(V(3)),
       S(V(3),2)
8026 FOR N=1 TO V(3)
8030 A(N)=0:B(N)=0:J(N)=0:M(N)=0:N(N)=0
8080 S(N,1)=0:S(N,2)=0
8090 NEXT N
8120 V=1234: RETURN
9010 PRINT "PROGRAM ENDED" : END
```

ROTATING CHRISTMAS DISPLAY

```
10 REM
20 REM
30 REM
40 REM
50 REM
60 REM  This is a jolly little display program that you can load up
70 REM  on your ELF, start it running, and just let it run continously
80 REM  on your CRT.
90 REM
100 REM It will make several different Christmas type pictures with
110 REM a pause between each.
120 REM
130 REM I usually set it up ready to run on Christmas morning and
140 REM and find it delights the visiting small fry.
150 REM
160 REM To stop the program, hit your BREAK key.
170 REM
180 REM
190 A=24:B=20:C=6
200 CLS: PRINT TAB(A);"M"
210 PRINT TAB(A-1);"ERR"
220 PRINT TAB(A-2);"Y CHR"
230 PRINT TAB(A-3);"ISTMAS"
240 PRINT TAB(A-4);"HAPPY NEW"
250 PRINT TAB(A-5);"YEAR MERRY"
260 PRINT TAB(A-6);"CHRISTMAS HAP"
270 PRINT TAB(A-7);"PY NEW YEAR MER"
280 PRINT TAB(A-8);"RY CHRISTMAS HAPP"
290 PRINT TAB(A-9);"Y NEW YEAR MERRY CH"
300 PRINT TAB(A-10);"RISTMAS HAPPY NEW YEAR"
310 PRINT TAB(A-11);"MERRY CHRISTMAS HAPPY NE"
320 PRINT TAB(A);"W"
330 PRINT TAB(A);"Y"
340 PRINT TAB(A-1);"EAR"
350 GOSUB 1040
360 PRINT TAB(B);"00000000"
370 PRINT TAB(B-5);"0000000000000000000"
380 PRINT TAB(B-8);"0000000                0000000"
390 PRINT TAB(B-10);"000000                  000000"
400 PRINT TAB(B-11);"000000                  000000"
410 PRINT TAB(B-11);"000000                  000000"
420 PRINT TAB(B-10);"000000                0000000"
430 PRINT TAB(B-8);"0000000              0000000"
440 PRINT TAB(B-9);"00000 0000000000000 00000"
450 PRINT TAB(B-10);"00000    00000000    00000"
460 PRINT TAB(B-11);"00000                00000"
470 PRINT TAB(B-10);"000                    000"
480 PRINT TAB(B-9);"0                        0"
490 GOSUB 1040
500 PRINT TAB(C);"M   M  EEEEEE  RRRRR   RRRRR   Y   Y"
510 PRINT TAB(C);"MM MM  E       R    R  R    R   Y Y"
520 PRINT TAB(C);"M M M  EEEEE   RRRRR   RRRRR    YY"
530 PRINT TAB(C);"M   M  E       R R     R R      YY"
540 PRINT TAB(C);"M   M  EEEEEE  R  R    R  R     YY"
```

```
550 PRINT : PRINT : PRINT : PRINT
560 PRINT TAB(C+2);"X      X   M     M      AA      SSSSSS"
570 PRINT TAB(C+3);"X    X     MM  MM     A  A     S"
580 PRINT TAB(C+4);"XXX      M MM M    AAAAAA    SSSSSS"
590 PRINT TAB(C+3);"X   X     M     M    A    A         S"
600 PRINT TAB(C+2);"X      X   X     X   A     A   SSSSSS"
610 GOSUB 1040
620 PRINT : PRINT : PRINT
630 PRINT TAB(C);"H    H   00000"
640 PRINT TAB(C);"H    H   O   O"
650 PRINT TAB(C);"HHHHH    O   O      H    H   00000"
660 PRINT TAB(C);"H    H   O   O      H    H   O   O"
670 PRINT TAB(C);"H    H   00000      HHHHH    O   O      H    H   00000"
680 D=C+18
690 PRINT TAB(D);"H    H   O   O      H    H   O   O"
700 PRINT TAB(D);"H    H   00000      HHHHH    O   O"
710 E=D+18
720 PRINT TAB(E);"H    H   O   O"
730 PRINT TAB(E);"H    H   00000"
740 GOSUB 1040
750 G=C+4
760 PRINT TAB(G);"H     H  AAAAAA  PPPPPP  PPPPPP  Y     Y"
770 PRINT TAB(G);"H     H  A     A  P     P  P     P  Y   Y"
780 PRINT TAB(G);"HHHHHHH  AAAAAA  PPPPPP  PPPPPP     YY"
790 PRINT TAB(G);"H     H  A     A  P        P           YY"
800 PRINT TAB(G);"H     H  A     A  P        P           YY"
810 PRINT : PRINT : PRINT :C=C-4
820 PRINT TAB(C);"N     N  EEEEEE  W     W     Y     Y  EEEEEE  AAAAAA  RRRRR"
830 PRINT TAB(C);"NN    N  E       W     W      Y   Y   E       A     A  R    R"
840 PRINT TAB(C);"N N   N  EEEEE   W WW W       YY     EEEEE   AAAAAA  RRRRR"
850 PRINT TAB(C);"N  N  N  E       WW  WW       YY     E       A     A  R   R"
860 PRINT TAB(C);"N   NN   EEEEEE  W     W      YY     EEEEEE  A     A  R    R"
870 GOSUB 1040
880 PRINT TAB(A);"!"
890 PRINT TAB(A-1);"( )"
900 PRINT TAB(A-2);"(   )"
910 PRINT TAB(A-1);"( )"
920 PRINT TAB(A);"!"
930 H=A-3
940 PRINT TAB(H);"000o000"
950 A$="0     0"
960 FOR N=1 TO 5
970 PRINT TAB(H);A$
980 NEXT N
990 PRINT TAB(H);A$;"          /"
1000 PRINT TAB(H);A$;"        /"
1010 PRINT TAB(H-11);"_____0_____0_____/"
1020 WAIT(700): GOTO 190
1030 END
1040 WAIT(700): CLS
1050 RETURN
```


23 QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

ADDRESS CORRECTION REQUESTED

#72194
[redacted]