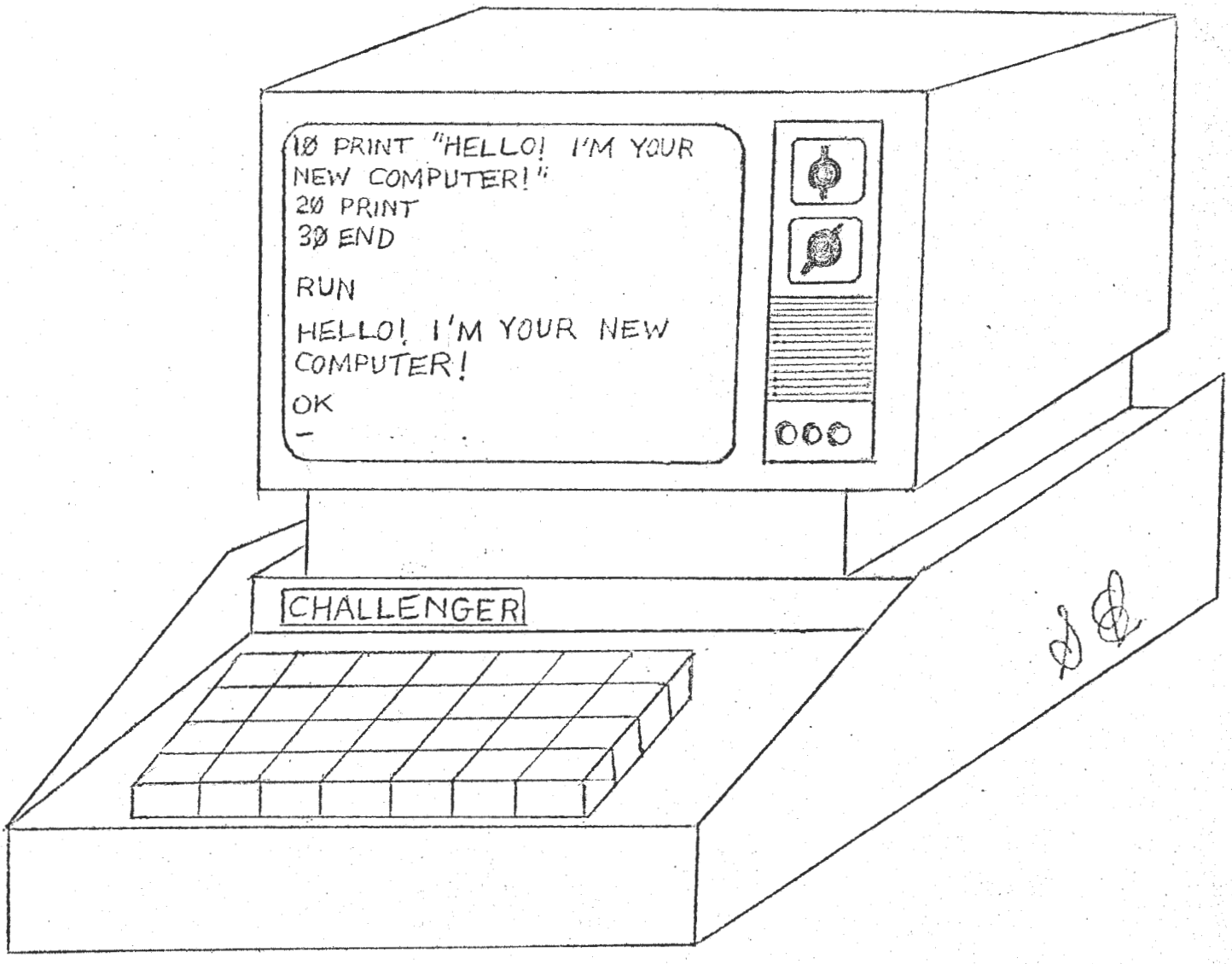
ISSUES

APRIL 1980
VOLUME 2
ISSUE 3



In this issue.....

Securing your programs
2 great demos
Speed up your superboard
C1 trivia quiz
..... and much more

OS·ITEMS VOLUME 2 ISSUE 3 April 1980

Edited by Salomon Lederman

40 Waterside Plaza, #31C

New York, NY 10010 (212) 685-9822

(On this month's cover is a great quote from OSI's users manual.******)

CONTENTS

Program Security

by Mike Bassman

This article will attempt to teach a little something about protecting your programs. For example, let's say that you want to load a program for someone on your system and you don't want them to see the program while it is loading. What can you do? Well, as you can imagine, in this article I will show you how to stop listings.

First let's look at the normal procedure for saving programs. First you type SAVE and the computer goes into the slow list mode. Then you put your recorder on the record mode and you hit return after having typed LIST. This is the normal procedure.

Now for the protective procedure. Type SAVE. Your system will go into the slow list mode. Now put your recorder in the record mode and type PRINT CHR$(15):LIST. Your recorder must record that statement and the program listing right after it. The effect; when you load the program it will not appear on the screen as it is being loaded.

But this is not a complete procedure because someone can list the program after it has loaded. A feasible solution to this problem is to make the program auto run. What this means is that as soon as the program finishes loading, it will automatically start to execute. This can be accomplished fairly simply by recording a RUN command on the tape right after your program is finished saving.

Of course no cassette program security system is foolproof but you might be able to discourage some people from stealing your ideas.

Danny's article, immediately following, presents another idea about program security.

Invisible Programs

by Danny Schwartz

Here is an interesting way to hide programs in memory, in other words to protect them from being listed from memory. The program is pretty straightforward. It uses the fact that when BASIC lists a program, it looks at pointers to the next line to be listed. If you change some pointers, then you can prevent certain parts of the program from being listed.

```
10 REM INVISIBLE PROGRAM
20 GOSUB 2000:REM PROTECTS THE LISTING
30 FOR X=1 TO 10:PRINTX:NEXT:REM SAMPLE PROGRAM LINE HIDDEN BUT ACTIVE
40 GOTO 50
45 STOP:REM IT SKIPS OVER THIS-DOESN'T STOP
50 REM DESTINATION OF GOTO/GOSUB/THEN MUST BE A REMARK
60 PRINT"IT GOT HERE":PRINT:LIST
2000 REM THIS ROUTINE WILL MAKE THE LINKAGE POINTER OF EACH
2010 REMARK STATEMENT POINT TO THE NEXT REMARK STATEMENT, SO ONLY
2020 REMARKS WILL BE LISTED
2030 P=769:SP=P:GOTO 2048
2038 REM
2040 IF PEEK(P+4)=142 THEN PH=INT(P/256):POKE SP+1,PH:POKE SP,P-256*
PH:SP=P
2048 REM
2050LP=PEEK(P)+256*PEEK(P+1):IFLPTHENP=LP:GOTO2038
2060 PH=INT(P/256):POKE SP+1,PH:POKESP,P-256*PH:RETURN
```

And to make this series complete Terry Terrance suggests that if you POKE 544,17 and POKE 545,189, then if you try to do a save BASIC will send you off to coldstart.

Poker Routine

by Mike Cohen

This neat little program will take those machine language routines that you've written and output them as DATA statements, complete with line numbers. SO... let's say that you've just written this great routine in machine language, and you'd like to use it as a user routine from BASIC. You feed the right information to this program, press RECORD on your recorder, and this program will mimic a listing of DATA statements, which can then be reloaded and SAVEd with a program

```
10 INPUT"STARTING LINE #";LN:INPUT"STARTING ADDRESS";SA
15 INPUT"ENDING ADDRESS";AE:INPUT"ITEMS PER LINE";ST
18 POKE 517,1
20 FOR A=SA TO AE STEP ST
30 PRINT LN;"DATA";
40 FOR B=0 TO ST-1
50 PRINT MID$(STR$(PEEK(A+B)),2);
55 IF B+A=AE THEN 80
60 IF B<ST-1 THEN PRINT",";
70 NEXT B:PRINT:LN=LN+10:NEXTA
80 PRINT:END
```

OSI

by Mike Bassman

Here's an animation demo, kind of long, but certainly worth typing in to your system.

```
1 REM ***OSI C1P ANIMATION DEMO***
2 REM *** Mike Bassman ***
10 FORK=53248TO54272:POKEK,32:NEXT
20 DATA210,135,135,207,128,209,129,128,208,208,220,223,220,223,32
30 L1=53732:GOTO70
40 FORR=1TO3:FORF=1TO5:READC:POKEL1+F+R*32,C:NEXTF:NEXTR
45 RETURN
50 FORK=1TO200:NEXTK:FORR=1TO3:FORF=1TO5:POKEL1+F+R*32,32:NEXTF:NEXTR
60 RETURN
70 FORG=1TO18:L1=L1+1:GOSUB40:GOSUB50:RESTORE:NEXTG:GOSUB40
80 POKEL1+32*3,195
```

```
90 FORK=1TO5:JOE(K)=L1+32*2:NEXTK
100 FORG=1TO5:POKEJOE(G),240:FORK=1TO18-G:D=-1:L=JOE(G)
110 GOSUB1000:JO(G)=JO(G)+D:FORR=1TO100:NEXTR:NEXTK:NEXTG
115 POKEL1+32*3,32
120 RESTORE:GOSUB50:FORW=1TO5:L1=L1+1:GOSUB40:GOSUB50:RESTORE:NEXTW
140 A$="CLUTZ BROS. CONSTRUCTION        COMPANY":GOSUB1010
150 D=32:FORK=1TO8:L=JO(1):GOSUB1000:JO(1)=JO(1)+D:FORR=1TO100
160 NEXTR:NEXTK
170 D=-1:FORK=1TO5:L=JO(1):GOSUB1000:JO(1)=JO(1)+D:FORR=1TO100:NEXTR
180 NEXTK:POKEJO(1)+1,154
190 D=1:FORK=1TO5:L=JO(1)+1:GOSUB1000:L=JO(1):GOSUB1000:JO(1)=JO(1)+1
200 FORR=1TO200:NEXTR:NEXTK
210 POKEJO(1)+1,32:POKEJO(1)-32,154
215 D=-32
220 FORK=1TO5:L=JO(1)-32:GOSUB1000:L=JO(1):GOSUB1000:JO(1)=JO(1)+D
230 FORR=1TO200:NEXTR:NEXTK
240 A$="THE CORNERSTONE":GOSUB1010
245 D=32
250 FORG=2TO5:FORK=1TO3+G:L=JO(G):GOSUB1000:JO(G)=JO(G)+D
260 FORR=1TO100:NEXTR:NEXTK:NEXTG
270 FORG=2TO5:D=-1:FORK=1TO5+G:L=JO(G):GOSUB1000:JO(G)=JO(G)+D
280 FORR=1TO100:NEXTR:NEXTK:NEXTG
290 FORK=2TO5:POKEJO(K)+1,136:NEXTK
300 D=1:FORG=2TO5:FORK=1TO7:L=JO(G)+1:GOSUB1000:L=JO(G)
310 GOSUB1000:JO(G)=JO(G)+D:FORR=1TO200:NEXTR:NEXTK:NEXTG
320 FORG=2TO5:POKEJO(G)+1,32:FORF=1TOG+2
321 GOSUB325:GOTO360
325 IFF=1ORF=5THENC=190:RETURN
330 IFF=2ORF=6THENC=132:RETURN
340 IFF=3ORF=7THENC=189:RETURN
350 IFF=4THENC=136:RETURN
355 RETURN
360 POKEJO(G)+1+-32*F,C:FORR=1TO25:NEXTR:POKEJO(G)+1+-32*F,32:NEXTF
370 X1=PEEK(JOE(1)-32):GOSUB380:GOTO420
380 IFX1=154THENX1=158:RETURN
390 IFX1=158THENX1=159:RETURN
400 IFX1=159THENX1=160:RETURN
410 IFX1=160THENX1=161:RETURN
420 POKEJO(1)-32,X1:NEXTG
430 D=-1:FORK=1TO5:L=JO(1):GOSUB1000:FORR=1TO100:NEXTR:JO(1)=JO(1)+D
435 FORG=1TO150:NEXTG
440 NEXTK:POKEJO(1)+33,161
445 D=32:L=JO(1):GOSUB1000:JO(1)=JO(1)+D:D=1
450 FORK=1TO6:L=JO(1)+1:GOSUB1000:L=JO(1):GOSUB1000:JO(1)=JO(1)+D
460 FORR=1TO250:NEXTR:NEXTK
470 A$="BOY,THIS IS HEAVY!":GOSUB1010
480 A$="LET ME HELP YOU":GOSUB1010
490 D=1:FORK=1TO2:L=JO(2):GOSUB1000:JO(2)=JO(2)+D:FORG=1TO100:NEXTG
500 NEXTK:D=-32:L=JO(2):GOSUB1000:JO(2)=JO(2)+D
505 FORK=1TO500:NEXTK
510 FORK=1TO7:L=JO(2)-1+(K-1)*D:GOSUB1000:FORG=1TOK*37:NEXTG
520 NEXTK:D=32
```

```
530 FORK=0TO6:GOSUB1000:FORG=1TO(6-K)*37:NEXTG:NEXTK
540 A$="SPLAT!!":GOSUB1010
550 A$="LETS TRY THAT AGAIN":GOSUB1010
555 BH=7:GOSUB560:GOTO530
560 D=-32:FORK=1TO10:GOSUB1000:FORG=1TOK*30:NEXTG:NEXTK:D=-1
570 FORK=1TO2:GOSUB1000:FORG=1TO200:NEXTG:NEXTK:D=32:FORK=1TOBH
572 GOSUB1000
575 FORG=1TO(6-K)*37:NEXTG:NEXTK:RETURN
580 A$="DONE!":GOSUB1010
585 FORG1=1TO5
590 D=-1:FORK=1TO5:L=JO(1):GOSUB1000:JO(1)=L:FORG=1TO200:NEXTG:NEXTK
600 POKEJO(1)+1,161:D=1
610 FORK=1TO5:L=JO(1)+1:GOSUB1000:L=L-2:GOSUB1000:JO(1)=L:FORG=1TO200
620 NEXTG:NEXTK
630 BH=BH-1:L=JO(1)+1:GOSUB560:NEXTG1:FORG=1TO100:NEXTG
640 D=32:L=JO(2):GOSUB1000:JO(2)=L:FORK=1TO2:D=-1:L=JO(2):GOSUB1000
645 FORE=1TO150:NEXTE
650 JO(2)=L:NEXTK:D=-1:FORK=1TO5:FORG1=1TO3
660 L=JO(G1):GOSUB1000:JO(G1)=L:FORK1=1TO100:NEXTK1:NEXTG1:NEXTK
670 D=1:FORK=1TO3:POKEJO(K)+1,161:NEXTK:FORG=1TO3:FORK=1TO4+G
680 L=JO(G)+1:GOSUB1000:L=JO(G):GOSUB1000:JO(G)=L:FORG1=1TO175:NEXTG1
690 NEXTK:NEXTG:FORK=1TO3:POKEJO(K)+1,32:POKEJO(K)-32,161:NEXTK:D=-32
700 FORG=2TO3:FORK=2TOG:L=JO(G)+D:GOSUB1000:L=JO(G):GOSUB1000:JO(G)=L
710 FORE=1TO200:NEXTE:NEXTK:NEXTG:BH=8
720 D=32:FORK=1TO2:L=JO(3):GOSUB1000:JO(3)=L:FORG=1TO100:NEXTG:NEXTK
730 D=1:FORK=1TO4:L=JO(2):GOSUB1000:JO(2)=L:L=JO(1):GOSUB1000:JO(1)=L
740 FORG=1TO100:NEXTG:NEXTK:D=1:L=JO(2):GOSUB1000:JO(2)=L:FORG=1TO150
750 NEXTG:FORG1=1TO7:D=-1:FORK=1TO8:L=JO(1):GOSUB1000:JO(1)=L
760 FORG=1TO100:NEXTG:NEXTK:POKEJO(1)+1,161:D=1:FORK=1TO3
770 L=JO(1)+1:GOSUB1000:L=JO(1):GOSUB1000:JO(1)=L:FORG=1TO200:NEXTG
780 NEXTK:L=JO(1)+1:GOSUB560:BH=BH-1:NEXTG1:D=-1
790 FORK=1TO2:L=JO(3):GOSUB1000:JO(3)=L:FORG=1TO100:NEXTG:NEXTK
800 D=1:FORK=1TO4:L=JO(2):GOSUB1000:JO(2)=L:L=JO(1):GOSUB1000
810 JO(1)=L:FORG=1TO100:NEXTG:NEXTK:D=-1:FORG=1TO5:FORK=3TO5
820 L=JO(K):GOSUB1000:JO(K)=L:FORG1=1TO100:NEXTG1:NEXTK:NEXTG
830 FORK=3TO5:POKEJO(K)+1,161:NEXTK:D=1:FORG=3TO5:FORK=1TO7+G
840 L=JO(G)+1:GOSUB1000:L=JO(G):GOSUB1000:JO(G)=L:FORG1=1TO100:NEXTG1
850 NEXTK:NEXTG:D=-32:FORK=3TO5:POKEJO(K)+1,32:POKEJO(K)+D,151:NEXTK
860 FORK=3TO5:FORG=1TOK-1:L=JO(K)+D:GOSUB1000:L=JO(K):GOSUB1000
870 JO(K)=L:FORG1=1TO175:NEXTG1:NEXTG:NEXTK:D=32:FORG=1TO5
880 FORK=3TO5:L=JO(K):GOSUB1000:JO(K)=L:FORG1=1TO100:NEXTG1:NEXTK
890 NEXTG:FORK=1TO5:L=JO(2):GOSUB1000:JO(2)=L:FORG=1TO100:NEXTG:NEXTK
900 A$="THERE MUST BE A BETTER          WAY TO DO THIS":GOSUB1010
905 D=-1:L1=53950:F=0:FORK=1TO14:GOSUB910:NEXTK:GOTO1030
910 POKEL1-32,176:POKEL1-31,161:POKEL1,225:POKEL1+1,225:FORG=2TO11
920 POKEL1-32*G+1,136:NEXTG:POKEL1-32*11,135:POKEL1-32*11-1,203
925 IFF=0THENGOSUB930
926 RETURN
930 FORG1=1TO100:NEXTG1:POKEL1-32,32:POKEL1-31,32:POKEL1,32
940 POKEL1+1,32:FORG=2TO11:POKEL1-32*G+1,32:NEXTG:POKEL1-32*11,32
950 POKEL1-32*11-1,32:L1=L1+D:RETURN
1000 X=PEEK(L):IFX=32THENSTOP
1005 POKEL,32:L=L+D:POKEL,X:RETURN
```

```
1010 FORK=1TOLEN(A$):POKE53413+K,ASC(MID$(A$,K,1)):NEXTK
1020 FORK=1TO2500:NEXTK:FORK=1TOLEN(A$):POKE53413+K,32:NEXTK:RETURN
1030 L1=L1-D:F=1:GOSUB910:A$="A CRANE !":GOSUB1010
1035 S=0:BH=9:GOSUB1037:GOTO1120
1037 FORG1=1TO7:D=-1
1040 FORG=1TO13+S:L=JO(1):GOSUB1000:JO(1)=L:FORK=1TO100:NEXTK:NEXTG
1045 D=1
1050 POKEJO(1)+1,161:FORG=1TO13+S:L=JO(1)+1:GOSUB1000:L=JO(1)
1060 GOSUB1000:JO(1)=L:FORE=1TO175:NEXTE:NEXTG
1070 E=L1-32*11-1:FORK=1TO10:POKEE+K*32,143:POKEE+(K+1)*32,203
1080 POKEE,143:FORG=1TO100:NEXTG:NEXTK:FORK=11TO1STEP-1
1090 D=-32:L=E+32*K:GOSUB1000:L=E+32*K-1:GOSUB1000:FORG=1TO100:NEXTG
1100 NEXTK:D=-1:GOSUB1000:FORK=1TOBH:D=32:GOSUB1000:FORG=1TO45:NEXTG
1110 NEXTK:BH=BH-1:NEXTG1:RETURN
1120 BH=9:S=9:D=1:GOSUB930:F=0:FORR=1TO6:GOSUB910:NEXTR
1130 L1=L1-D:F=1:GOSUB910:FORR=1TO6:L=JO(1):GOSUB1000:JO(1)=L
1140 FORK=1TO100:NEXTK:NEXTR:GOSUB1037
1150 D=1:F=0:GOSUB930:FORR=1TO2:GOSUB910:NEXTR:F=1:GOSUB910
1155 F=0:L3=L1:D=32:FORK=1TO5
1160 L=JO(1):GOSUB1000:JO(1)=L:FORG=1TO100:NEXTG:NEXTK:D=-32
1170 FORK=3TO5:FORG=1TOK-1:L=JO(K):GOSUB1000:JO(K)=L:FORR=1TO100:NEXTR
1180 NEXTG:NEXTK:D=-1:FORG=3TO5:FORK=1TO8+G:L=JO(G):GOSUB1000
1190 JO(G)=L:FORR=1TO100:NEXTR:NEXTK:NEXTG:D=1:FORK=3TO5
1200 POKEJO(K)+1,161:NEXTK:FORG=3TO5:FORK=1TO15+G:L=JO(G)+1:GOSUB1000
1210 L=JO(G):GOSUB1000:JO(G)=L:FORR=1TO175:NEXTR:NEXTK:NEXTG
1220 FORK=3TO5:POKEJO(K)+1,32:POKEJO(K)-32,161:NEXTK:D=-32
1230 FORK=3TO5:FORG=1TO6-K:L=JO(K)+D:GOSUB1000:L=JO(K):GOSUB1000
1240 JO(K)=L:FORR=1TO175:NEXTR:NEXTG:NEXTK
1250 RESTORE:FORK=1TO15:READA:NEXTK:D=1:L2=53477:POKEL2,237
1260 FORG1=1TO9:READA:GOTO1290
1265 D=32
1270 POKEL2+32,161:FORK=2TO5:L=L2+32*(K-1):GOSUB1000:FORR=1TO35:NEXTR
1280 NEXTK:D=1:GOTO1300
1290 IFL2=A-32*5THEN1265
1300 L=L2:GOSUB1000:L2=L
1310 IFL2=53499THENPOKEL2,32:L2=53477:POKEL2,237:NEXTG1:GOTO2000
1320 FORR=1TO75:NEXTR:GOTO1290
1330 DATA53638,53639,53640,53647,53648,53649,53651,53652,53653
2000 POKEL2,32:D=-32:FORK=1TO5:L=JO(2):GOSUB1000:JO(2)=L:FORG=1TO100
2010 NEXTG:NEXTK:D=1:FORK=1TO2:L=JO(2):GOSUB1000:JO(2)=L:FORG=1TO100
2020 NEXTG:NEXTK:D=-1:FORK=1TO3:L=JO(1):GOSUB1000:JO(1)=L:FORG=1TO100
2030 NEXTG:NEXTK:D=-32:FORK=1TO5:L=JO(1):GOSUB1000:JO(1)=L:FORG=1TO100
2040 NEXTG:NEXTK:L1=54020:RESTORE
2050 FORG=1TO18:L1=L1+1:GOSUB40:GOSUB50:RESTORE:NEXTG:GOSUB40
2060 POKEL1+32*3,195
2070 FORG=1TO5:D=32:FORK=1TO5:L=JO(G):GOSUB1000:JO(G)=L:FORG1=1TO100
2080 NEXTG1:NEXTK:D=1:FORK=1TO6-G:L=JO(G):GOSUB1000:JO(G)=L
2090 FORG1=1TO100:NEXTG1:NEXTK:POKEJO(G),32:NEXTG
2095 POKEL1+32*3,32
2100 RESTORE:GOSUB50:FORG=1TO6:L1=L1+1:GOSUB40:GOSUB50:RESTORE:NEXTG
2110 L1=L3:D=1:F=0:GOSUB930:FORR=1TO4:GOSUB910:NEXTR
2120 FORK=1TO20000:NEXTK:RUN
OK
```

Hex Fifteen Puzzle

by Danny Schwartz

Here's a really nice program for C1 people. It simulates the fifteen puzzle, in which you have to slide numbers around and get them in order. The program will ignore all invalid moves, and when you enter a number it figures out which way to slide it. The program accepts the numbers from 1 through 9 and the letters A through F to move a piece. There are no provisions for a win but an enterprising programmer could add that in.

```
5 DIM C(16),V(16),L(15)
6 H$="123456789ABCDEF "
7 POKE 11,0:POKE 12,253
10 PRINT:PRINT"HEX FIFTEEN PUZZLE"
30 FOR X=1 TO 16
40 C=INT(16*RND(1)):IF L(C) THEN 40
50 C(X)=C:L(C)=X
60 NEXT X
65 FOR X=1 TO 10:PRINT:NEXT
70 FORY=0 TO 3:FOR X=1 TO 4
72 V(4*Y+X)=53700+4*X+64*Y
75 C=C(4*Y+X)
76 IF C=0 THEN C=16
80 PRINT"   ";MID$(H$,C,1);
90 NEXT X:PRINT:PRINT:NEXTY
100 PRINT:PRINT:PRINT:PRINT:PRINT
110 X=USR(X):I=PEEK(531):IF I<48 OR I>70 OR I>57 AND I<65 THEN 110
120 C=I-48+7*(I>60)
130 L=L(C):Y=INT((L-1)/4):X=L-1-4*Y
140 IF X=0 THEN 150
145 IF C(L-1)=0 THEN IN=-1:GOTO 200
```

Press Release

by Salomon Lederman

Over two years in the working... by a team of expert programmers ... created by the people who brought you MICROSOFT BASIC, SARGON for the APPLE, and high resolution SPACE INVADERS for the APPLE... to be sold exclusively by Antfarm Technical Services... we are extremely proud to introduce... SUPER MACRO DELUXE HI-LO EXTENDED VERSION 1.0, REVISION 3.2.

That's right, the same great game that everyone thought would run only on large timesharing systems is now available for the C8 with color, sound, and graphics. Be wary of cheap imitations. Due to recent breakthroughs in the field of programming theory our team of over two dozen MIT graduates has discovered the optimum algorithm for handling all of the complex operations in each turn, with greater speed and accuracy than ever before. Never again wait ten to fifteen seconds while some large IBM 370 tries to determine the clue for your next turn. Now you can merely turn on your C8, load the disk, and youre all set for hours of nonstop fun.

You read correctly. Only a single disk drive needed to run this fabulous HI-LO program; and with a mere 32K you can run this program with great features. Four of the top artists in the United States have created a unique layout for the incredible graphics. Sound and color add life to the game. Over twelve levels of difficulties. Special routines allow you to save games to continue at a future date. Will allow up to four players to compete with each other. Automatically updates the score for you. And best of all, deeply buried in BASIC, we have discovered a unique guess input routine. Merely enter your guess after the system produces a question mark. This truly ingenuous

program handles the rest. There are over six error codes to keep you on the right track.

But best of all SUPER MACRO DELUXE HI-LO EXTENDED VERSION 1.0, REVISION 3.2 is fast. That's because it's in machine language.

So... run out and buy a copy as soon as SUPER MACRO DELUXE HI-LO EXTENDED VERSION 1.0, REVISION 3.2 is released to your favorite dealer. And if you act now and send a deposit to Antfarm Technical Services, we'll send, with your first order, absolutely free of charge, our 64 page guide to playing HI-LO. So order now before the prices go up; only $49.95 for one or $112.95 for two copies of the game.

Joysticks

by Larry Thaler

I recently purchased a pair of joysticks for my C1 from Aurora Software Associates (353 South 100 East, #6 Springville, Utah 84663). The pair cost $24.95 and took about a month to get to me.

It was well worth waiting. What I received was a small box containing 2 grade A #1 joysticks (similar to those used in Atari video games) and very well laid out instructions. In order to use the joysticks, two modifications had to be made by passing a couple of diodes in the keyboard circuitry. The instructions were clear and specific, and after doing the mod and plugging in the joysticks, I was pleased to note that everything worked fine. There is one small mistake however in the sample program which comes with the joysticks; several modifications should be made as follows:

change line 600 to IF P=255 THEN M=0:RETURN

and add 685 IF P=251 THEN POKE 530,0:STOP

Line 600 will allow the two tanks to move independently, and line 685 will let you break out of the program by pressing RUBOUT.

Reflex Test

by Mike Bassman

This program is fairly selfexplanatory in nature. The computer randomly picks letters and you have to type them in as fast as possible. You get ten keys to press and then the computer gives you your own personal evaluation on how you did.

```
1 REM ***REFLEX TEST***
2 REM *** Mike Bassman ***
3 FORK=1TO35:PRINT:NEXT
5 INPUT"DO YOU WANT INSTRUCTIONS";Y$
6 IFLEFT$(Y$,1)<>"Y"THEN 120
10 FORK=53248TO54272:POKEK,32:NEXT
20 PRINT"      REFLEX"
30 PRINT"-----------------"
40 PRINT:PRINT:PRINT:PRINT"A GAME OF REFLEXES"
50 FORK=1TO4000:NEXT
60 FORK=1TO25:PRINT:NEXT
70 PRINT"YOU WILL BE GIVEN":PRINT
80 PRINT"10 CHANCES AT HITTING":PRINT
```

```
 90 PRINT"10 DIFFERENT KEYS":PRINT
 100 PRINT"THE FASTER YOU HIT,":PRINT
 110 PRINT"THE BETTER YOUR SCORE"
 115 FORK=1TO4000:NEXT
 120 FORK=53248TO54272:POKEK,32:NEXT
 122 POKE530,1
 124 DIM XX(10,2)
 126 FAS=20000:SLO=-1
 130 P=57088
 140 DIMS$(3,7)
 150 DATA"W","E","R","T","Y","U","I","S","D","F","G","H"
 153 DATA"J","K","X","C","V","B","N","M",","
 155 FORG=1TO3:FORK=1TO7
 160 READ Q$:S$(G,K)=Q$
 170 NEXTK:NEXTG
 175 RESTORE
 180 FORG=1TO10
 190 ROW=INT(RND(1)*3+1):COLUMN=INT(RND(1)*7+1)
 195 IF RO=R1ANDCO=C1THEN 190
 197 R1=RO:C1=CO
 200 IF ROW=1THENPOKEP,239
 210 IF ROW=2THENPOKEP,247
 220 IF ROW=3THENPOKEP,251
 230 ON COLUMN GOTO 240,250,260,270,230,230,300
 240 K=127:GOTO310
 250 K=191:GOTO310
 260 K=223:GOTO310
 270 K=239:GOTO310
 280 K=247:GOTO310
 290 K=251:GOTO310
 300 K=253
 310 PRINT"HIT THE ";S$(ROW,COLUMN);" KEY"
 320 IFPEEK(P)=KTHEN330
 325 Z=Z+1:GOTO320
 330 PRINT"REFLEX TIME WAS";Z:S=S+Z
 332 XX(G,1)=Z:XX(G,2)=ASC(S$(ROW,COLUMN))
 335 PRINT
 336 IFZ> SLOTHEN SLO=Z
 337 IFZ< FAS THEN FAS=Z
 338 Z=0
 340 NEXTG
 342 FORG=1TO20:PRINT:NEXT
 350 PRINT"AVERAGE SPEED WAS";S/10:S=0
 353 PRINT:PRINT"FASTEST RESPONSE WAS";FAS
 355 PRINT:PRINT"SLOWEST RESPONSE WAS";SLO
 356 INPUT"SAY YES FOR A COMPLETE   RUNDOWN";Y$
 357 IFLEFT$(Y$,1)<>"Y"THEN 359
 358 FORG=1TO10:PRINT"(";CHR$(XX(G,2));")---";XX(G,1):NEXTG
 359 PRINT:PRINT
 360 FORK=1TO1000:NEXT
 370 CLEAR:INPUT"TRY AGAIN";Y$:IFLEFT$(Y$,1)="Y"THEN 120
OK
```

```
 90 PRINT"10 DIFFERENT KEYS":PRINT
 100 PRINT"THE FASTER YOU HIT,":PRINT
 110 PRINT"THE BETTER YOUR SCORE"
 115 FORK=1TO4000:NEXT
 120 FORK=53248TO54272:POKEK,32:NEXT
 122 POKE530,1
 124 DIM XX(10,2)
 126 FAS=20000:SLO=-1
 130 P=57088
 140 DIMS$(3,7)
 150 DATA"W","E","R","T","Y","U","I","S","D","F","G","H"
 153 DATA"J","K","X","C","V","B","N","M",","
 155 FORG=1TO3:FORK=1TO7
 160 READ Q$:S$(G,K)=Q$
 170 NEXTK:NEXTG
 175 RESTORE
 180 FORG=1TO10
 190 ROW=INT(RND(1)*3+1):COLUMN=INT(RND(1)*7+1)
 195 IF RO=R1ANDCO=C1THEN 190
 197 R1=RO:C1=CO
 200 IF ROW=1THENPOKEP,239
 210 IF ROW=2THENPOKEP,247
 220 IF ROW=3THENPOKEP,251
 230 ON COLUMN GOTO 240,250,260,270,230,230,300
 240 K=127:GOTO310
 250 K=191:GOTO310
 260 K=223:GOTO310
 270 K=239:GOTO310
 280 K=247:GOTO310
 290 K=251:GOTO310
 300 K=253
 310 PRINT"HIT THE ";S$(ROW,COLUMN);" KEY"
 320 IFPEEK(P)=KTHEN330
 325 Z=Z+1:GOTO320
 330 PRINT"REFLEX TIME WAS";Z:S=S+Z
 332 XX(G,1)=Z:XX(G,2)=ASC(S$(ROW,COLUMN))
 335 PRINT
 336 IFZ> SLOTHEN SLO=Z
 337 IFZ< FAS THEN FAS=Z
 338 Z=0
 340 NEXTG
 342 FORG=1TO20:PRINT:NEXT
 350 PRINT"AVERAGE SPEED WAS";S/10:S=0
 353 PRINT:PRINT"FASTEST RESPONSE WAS";FAS
 355 PRINT:PRINT"SLOWEST RESPONSE WAS";SLO
 356 INPUT"SAY YES FOR A COMPLETE   RUNDOWN";Y$
 357 IFLEFT$(Y$,1)<>"Y"THEN 359
 358 FORG=1TO10:PRINT"(";CHR$(XX(G,2));")---";XX(G,1):NEXTG
 359 PRINT:PRINT
 360 FORK=1TO1000:NEXT
 370 CLEAR:INPUT"TRY AGAIN";Y$:IFLEFT$(Y$,1)="Y"THEN 120
OK
```

OSI Challenger 1P Trivia Quiz

by Salomon Lederman

I wrote this trivia quiz with the idea that maybe people would learn something about their C1's, and perhaps people could collect 'trivial' pokes and peeks and publish them in OS-ITEMS so that we all could learn about our C1, C2, C4,...etc. What may seem trivial to one person may be something that someone else has been trying to figure out for months. So maybe you'll learn something about your C1. Or maybe you're an expert. Take this quiz and find out. (And don't use your C1, that's cheating.) All answers can be either in HEX or decimal.

1. What is the default value if you hit return for TERMINAL WIDTH from coldstart?
2. How many keys are there on a C1 keyboard?(Don't peek.)
3. According to the C1 what is the value of (25 AND 46)?
4. What is the default value for MEMORY SIZE on an 8K system?
5. If K=0 what will a C1 respond to PRINT K=0 ?
6. What is the value of the graphic character that prints the cursor?
7. At what memory location is the terminal width stored?
8. Normally location 0000 points to A274. Where does 0003 point to?
9. What happens when you press CONTROL M?
10. At what location does a BASIC program usually begin?(the text.)
11. What is the coldstart adress?
12. To how many bits of accuracy are variables in BASIC stored?

continued

13. How many letters of a variable does BASIC recognize?

14. What unit of measurement does the C1 use to compute trigonometric functions?

15. What is the highest allowed line number for a BASIC program?

16. What do you get if you type A for D/C/W/M ?

17. On what page of memory is the stack located?

18. Which line number would use up more memory, 0015 or 750?

19. How many bytes does the keyword GOTO use in memory?

20. What is the normal position of the cursor (like after BASIC prints OK)?

Answer Key for Trivia Quiz; (Do the quiz before you read this.)
1. 72 2. 53 3. 8 4. 7423 bytes 5. -1 6. 95 7. 15 8. A8C3
9. same as hitting the RETURN key 10. 769, or 0301HEX 11. BD11
12. 24 13. 2 14. radians (180xPI radians=180 degrees, where PI=3.141592654) 15. 63999 16. nothing. However, if you type A for memory size you get "WRITTEN BY RICHARD W. WEILAND" 17. 1 18.neither, all line numbers are converted to 2 byte binary format
19. just one, all keywords are tokenized into 1 byte codes 20.54117

Scoring; Give yourself 5 points for each one you got right.
0-5: not very good at all, you'd better look into your C1 some more.
6-10: not too good, talk to some C1 owners, you could learn alot
11-15: pretty good, you seem to know quite a bit, share with others
16-20: fantastic, you ought to teach us some PEEKS and POKES

Update: people offering software, hardware, plans...

by Mike Bassman

J.L. Smith 11201 Crestfield Dr. Huntsville, AL 35803	PEEK(65)	software
S. Hoffman 873 Dorset Dr. Knoxville, TN 37919	PEEK(65)	C1P color graphics plans
James Beneke 12 Kent Dr. Orchard Park, NY 14127	PEEK(65)	keyboard Macro Utility
Stan New 7236 South Sedalia Str. Aurora, CO 80016	PEEK(65)	software
S. Chaflin 905 Clinton Str. Philadelphia, PA 19107	MICRO- March	C1P 50x30 video upgrade
Joseph Endre (Progrssive Computing) 3336 Avondale Court Windsor, Ontario CANADA N9E 1X6	MICRO- March	software
Personal & Business Computer Connection 38437 Grand River Farmington Hills, MI 48018	MICRO- March	hardware mods

A Truly 'Super' Superboard

by Larry Thaler

You're not going to believe this one! Yes, Mr. Jack Robert Swindell has done it again. From the man who brought us the double dissasembler (Micro 19:31), comes the great Superboard speed up.

Mr. Swindell's article, which appeared in Micro Magazine (Issue 21, page 31), describes a very simple modification to make the Superboard or the Challenger 1 twice as fast. Well, I'll try anything. I did and it did! Did what?... It worked!!!

My computer is now operating at twice its old speed, but due to some strange occurences, BASIC is operating at nearly four times its normal speed. (Try typing FOR X=1 TO 10000:NEXT (RETURN). Count off 30 seconds and use CONTROL C. Then PRINT X. My computer does 380 counts per second, and 1290 in sped-up mode.

This mod is really worth doing because all the bad programs that I've bought now operate at decent speeds and are much better.

Now that you know what the modification does I'll explain how it works. The Superboard, like any other computer, works off a set of signals known as clocking pulses. These pulses are generated in the clock generation section of the Superboard. It is then stepped down from 14 MHZ (14 million pulses per second) to 1 MHZ by a series of counters. This 1 MHZ wave is then distributed all over the computer to keep it all operating at the same speed. Well, if we bypass one of these counters the clockpulse is then raised to 2MHZ, making the computer work twice as fast.

I don't want to actually go through the mod here; it wouldn't be fair. Micro is a great magazine, and since you all ought to go

out and buy it anyway, I'll let you read it yourself. Mr. Swindell's instructions are accurate and well explained so that you should have no trouble following his directions.

A few other points; As stated in Micro , the modification does not affect the cassette port- it still operates as normal. Also, the mod will probably totally screw up your system if you are using a real time clock or a minifloppy system. I installed a switch to change between fast and normal mode. Thus I can run programs at their normal speed as well. DO NOT switch rates with the power on.

As Jack warns in Micro, if you have never handled a soldering iron, by all means don't do the mod. Even if you are not interested in doing the mod, read the article anyway. You will certainly learn something.

Note: In case you're curious, all but two of my RAM chips worked, and all my ROMS were fine.

Working Around Strings

by Mike Bassman

OSI 8K BASIC in ROM is very good in general, but there is a tragic flaw in the string handling routine. Specifically, when a string is changed or manipulated in any way, the string routine, rather than replacing a string with a new one, will create an entirely new string. Therefore, any program which uses strings extensively, such as a text editor, will consume huge amounts of memory in short order. This problem is compounded in string arrays, because BASIC must keep up with string memory demands in a more in-

flexible array structure. Effectively, string arrays in their present form cannot be used to any serious degree. This article will attempt to explain the solution to the problem. The answer is to store strings manually rather than using functions provided by BASIC. In other words, put each string, character by character, into memory as their ASCII equivalents. Each string must have its length preceding it so as to be able to distinguish between strings.

Example: 15 Now is the time

@memloc @memloc+15

To the computer this would look like (in HEX):

0F 4E 6F 77 20 69 73 20 74 68 65 20 74 69 60 65

Bookkeeping variables are also required. Specifically, we need a variable designating the beginning of memory and the current memory, i.e.- the beginning of memory is where we start to store strings and current memory location is where our next string will begin. Since (BASIC handled) strings are also stored in upper memory, where we wish to put our material, we must reset the limit of upper memory, initially typed in at coldstart. This limit is stored at 133 and 144 decimal, in lo,hi-byte format. This limit should be set right above program and variable memory requirements. On the following page is a program which shows these techniques. It will let you enter strings, list ones already entered, and let you delete the last one entered. Following the program is a fairly detailed explanation of the workings of the program.

```
LIST

 5 FORK=53248TO54272:POKEK,32:NEXT
 10 POKE133,0:POKE134,8:BM=2049:ML=2049:POKE11,0:POKE12,253
 20 PRINT"1 Enter a string":PRINT"2 List all entries"
 25 PRINT"3 Delete last entry"
 30 INPUTC:IFC<1ORC>3THENPRINT:GOTO20
 40 ONCGOTO70,100,200
 50 FORK=1TO27:PRINT:NEXT
 60 GOTO20
 70 PRINT"enter the string":INPUTA$
 80 POKEML,LEN(A$):FORK=1TOLEN(A$):POKEML+K,ASC(MID$(A$,K,1)):NEXTK
 90 ML=ML+LEN(A$)+1:GOTO50
 100 TL=BM:IFBM=MLTHEN50
 110 FORK=TL+1TOTL+PEEK(TL):PRINTCHR$(PEEK(K));:NEXTK:PRINT
 120 TL=TL+PEEK(TL)+1:IFTL<>MLTHEN110
 130 PRINT:PRINT"Hit cr to return":X=USR(X):GOTO50
 200 TL=BM:IFBM=MLTHEN50
 210 IFTL=ML-(PEEK(TL)+1)THENML=TL:GOTO50
 220 TL=TL+PEEK(TL)+1:GOTO210
OK
```

5 Clear screen.

10 Poke upper memory limit, initialize variables: Beginning of memory

20-25 Print choices.

30 Input choice, check for legality.

40 Branch to appropriate section of program, depending on choice.

50 Clear screen.

60 Branch back to menu.

70 Input a string from the keyboard.

80 Poke in length of string, poke in string, character by character.

90 Increment Current Memory Location(ML), go back to menu.

100 Set Temporary Memory Location(TL) to beginning of memory(BM). If there are no strings to list then branch back to menu.

110 Loop which prints out a string, using its length to judge loop parameters.

120 Increment TL, if we are not at end of file then go back to printing loop.

130 Wait until user is ready, then return to menu.

200 Set TL to BM, if there is no string to delete then return to menu.

210 If this is the last string, then decrement the Current Memory variable

220 Increment TL, go back to Last string check.

***If you learn these techniques, more advanced programs, including Text and Word processing, are now within reach.

contributed by Terry Terrance
(not originated here)

Resequence for O.S.I.

```
60000 END
60010 T=0:DIMV(100),W(100):GOSUB60160:FORR=1TO1E3:GOSUB60210
60020 IFGTHENGOSUB60090:NEXTR
60030 GOSUB60160:FORR=1TO1E3:N=INT(M/256):POKEA-1,M-N*256
60040 POKEA,N:U=L:GOSUB60070:W(J)=M:GOSUB60170:IFGTHENNEXTR
60050 GOSUB60160:FORR=1TO1E3:GOSUB60210:IFGTHENGOSUB60110:NEXTR
60060 ?"*END":END
60070 J=0:IFT<>0THENFORJ=1TOT:IFV(J)<>UTHENNEXTJ:J=0
60080 RETURN
60090 IFU<>0THENGOSUB60070:IFJ=0THENT=T+1:V(T)=U
60100 RETURN
60110 GOSUB60070:IFJ=0THENRETURN
60120 Z=W(J):IFZ=0THEN?"GO";U;"L";L;"?":RETURN
60130 FORD=ATOB+1STEP-1:X=INT(Z/10):Y=Z-10*X+48:IFZ=0THENY=32
60140 POKED,Y:Z=X:NEXTD:IFZ=0THENRETURN
60150 ?"INSERT";W(J);"L";L:RETURN
60160 F=769:M=90
60170 A=F:M=M+10
60180 F=PEEK(A)+PEEK(A+1)*256:L=PEEK(A+2)+PEEK(A+3)*256:A=A+3:G=L<6E4
60190 RETURN
60200 S=0
60210 U=0:A=A+1:B=A:C=PEEK(A):IFC=0THENGOSUB60170:ONG+2GOTO60210,60190
```

```
60220  IFC<>136ANDC<>140ANDC<>160ANDC<>5GOTO60200
60230  A=A+1:C=PEEK(A)-48:IFC=-16GOTO60230
60240  IFC>=0ANDC<=9THENU=U*10+C:GOTO60230
60250  S=44:A=A+1:RETURN
```

The largest of the lines in this program just fit into the line buffer so be sure to enter them exactly as written, no spaces.

Resequence is a nifty program that will renumber your programs by 10 starting with 100. To work, Resequence must sit behind your program (ergo the 60,000+ line numbering) since it cannot renumber itself without crashing.

After loading, evoke Resequence by RUN 60010. If the renumbering is totally sucessful you will see the message *END*. There are two other types of messages you may see before the end message.

One of them looks like this:

INSERT 120 L 100

This message is telling you that it was not able to insert the new line number in a line containing a GOTO,IF,etc. It has now given this task to you. You must insert the first number (the new line number which is the object of the branch) into the expression now at the new line number given by the second number. This occurs because the line numbers, which are objects of branches, are stored as ASCII characters. If the old line number was two digit and the new one is three digit, there is no space for the extra digit; without rearranging the BASIC line pointers, a formidable (and slow) task for a BASIC program. No such problem exists for the actual number of a line because that number is stored in two bytes in a product and sum format that can

address more than 64K lines.

The other type of message looks like this:

GO 40 L 150

What this is telling you is that the program has found a reference to a line 40 in the object program where no line 40 was found. Again you have to take corrective action and find out what is going on.

Watch out for Resequence, do not use it on a program that you do not yet have stored on tape. It can bomb, leaving you with a partially renumbered program or without line references corrected or listed. One cause of wipe-out can be exceeding the allowable number of line references (100 as written here).

If you want to be able to specify the starting line number and the increment, try adding

```
60005  ?: INPUT "FIRST LINE #"; LN:?: INPUT "INCREMENT"; IN
```

and change

```
60160  F = 769: M = LN - IN
60170  A = F  : M = M  + IN
```

I have not tried these changes but they should work. The program is now evoked by typing RUN 60005.

Elevator Demo

by Mike Bassman

This program is a very cute demo. Using the OSI graphics Mike has created an elevator, which rides up and down its shaft, picking up passengers and dropping people off. Press any of the numbers from 1 to 5 to call the elevator from a respective floor.

```
1 REM ***ELEVATOR***
2 REM *** Mike Bassman ***
10 FORK=53248TO54272:POKEK,32:NEXTK
20 X=53413
30 POKEX,210:POKEX+4,207:POKE54117,209:POKE54121,203
40 FORK=X+1TOX+3:POKEK,135:POKEK+22*32,128:NEXTK
50 FORK=X+32TOX+32*21STEP32:POKEK,136:POKEK+4,143:NEXTK
60 X=54022
70 FORK=1TO5:POKEX,ASC(MID$(STR$(K),2,1)):X=X-32*4:NEXTK
80 X=54024:FORK=1TO5:POKEX,226:X=X-32*4:NEXTK
90 FORK=53387TO53387+23*32STEP32:POKEK,151:NEXTK
100 X=54028
110 FORK=1TO5
120 FORG=1TO6:POKEX+G-1,134:NEXTG
130 FORG=1TO5:POKEX+G+10,184:NEXTG
140 X=X-32*4:NEXTK
150 X=53426
160 PC=0
170 D=32
180 FX=X
190 GOTO300
200 FORA1=1TO4:FORA2=FXTOFX+4:POKEA2,32:NEXTA2:FX=FX+32:NEXTA1
210 FORA1=53396TOX-30STEP32:POKEA1,143:NEXTA1
220 POKEX,210:POKEX+4,207:POKEX+32*3,209:POKEX+32*3+4,203
230 FORG=X+1TOX+3:POKEG,135:POKEG+32*3,134:NEXTG
240 POKEX-30,193:POKEX+33,241
250 IFPC=0THEN270
```

```
260 FORK=1TOPC:POKEX+68-K,240:NEXTK
270 FORK=X+32TOX+64STEP32:POKEK,136:POKEK+4,143:NEXTK
275 FORK=1TO100:NEXTK
280 RETURN
300 GOSUB200
310 GOSUB320:GOTO410
320 FORG=1TO5
330 IFINT(RND(1)*15+1)<>14THEN400
335 S2=0
340 FL=54157-(G*4*32)-32
350 FC(G)=FC(G)+1:IFFC(G)=4THENFC(G)=3:GOTO400
360 POKEFL-1,32:POKEFL,240
361 S2=S2+1
362 FORA2=1TO10:NEXTA2
365 IFS2=50RPEEK(FL+1)<>32THEN400
370 FL=FL+1:GOTO360
400 NEXTG:RETURN
410 FX=X:X=X+D:IFX=53394ORX=54066THEND=D*-1:X=X+D
420 GOSUB200
430 IFPEEK(X+95)=184ANDPEEK(X+86)=232THENGOSUB450
440 GOTO600
450 POKEX+68,32:FORK=1TO500:NEXTK
455 IFPC=0THEN490
460 FORG=1TOINT(RND(2)*PC+1)
465 PC=PC-1
470 POKEX+68-G,32:FORK=1TO6
480 POKEX+67+K,240:FORA1=1TO100:NEXTA1:POKEX+67+K,32:NEXTK:NEXTG
490 POKEX+68,143:FORA1=1TO500:NEXTA1
495 IFPC=0THEN520
500 FORK=X+65TOX+67:POKEK,32:NEXTK
510 FORK=1TOPC:POKEX+68-K,240:NEXTK
520 POKEX+64,32:FORK=1TO500:NEXTK
525 G1=VAL(CHR$(PEEK(X+34)))
527 IFFC(G1)=0THEN570
530 FORG=1TOFC(G1):SC=0
532 IFPC=3THEN560
535 FC(G1)=FC(G1)-1:PC=PC+1
540 POKEX+64-G+SC,32:POKEX+64-G+SC+1,240:SC=SC+1
545 FORA2=1TO100:NEXTA2
550 IFPEEK(X+64-G+SC+1)=32THEN540
560 NEXTG:POKEX+64,136:FORK=1TO500:NEXTK
570 POKEX+86,226
580 FORK=X+63TOX+60STEP-1:POKEK,32:NEXTK
585 IFFC(G1)=0THEN595
590 FORK=1TOFC(G1):POKEX+64-K,240:NEXTK
595 RETURN
600 POKE530,1:POKE57088,127
610 IFPEEK(57088)=127THENPOKE54024,232
620 IFPEEK(57088)=191THENPOKE54024-32*4,232
630 IFPEEK(57088)=223THENPOKE53768,232
640 IFPEEK(57088)=239THENPOKE53768-32*4,232
650 IFPEEK(57088)=247THENPOKE53512,232
660 GOTO310
OK
```