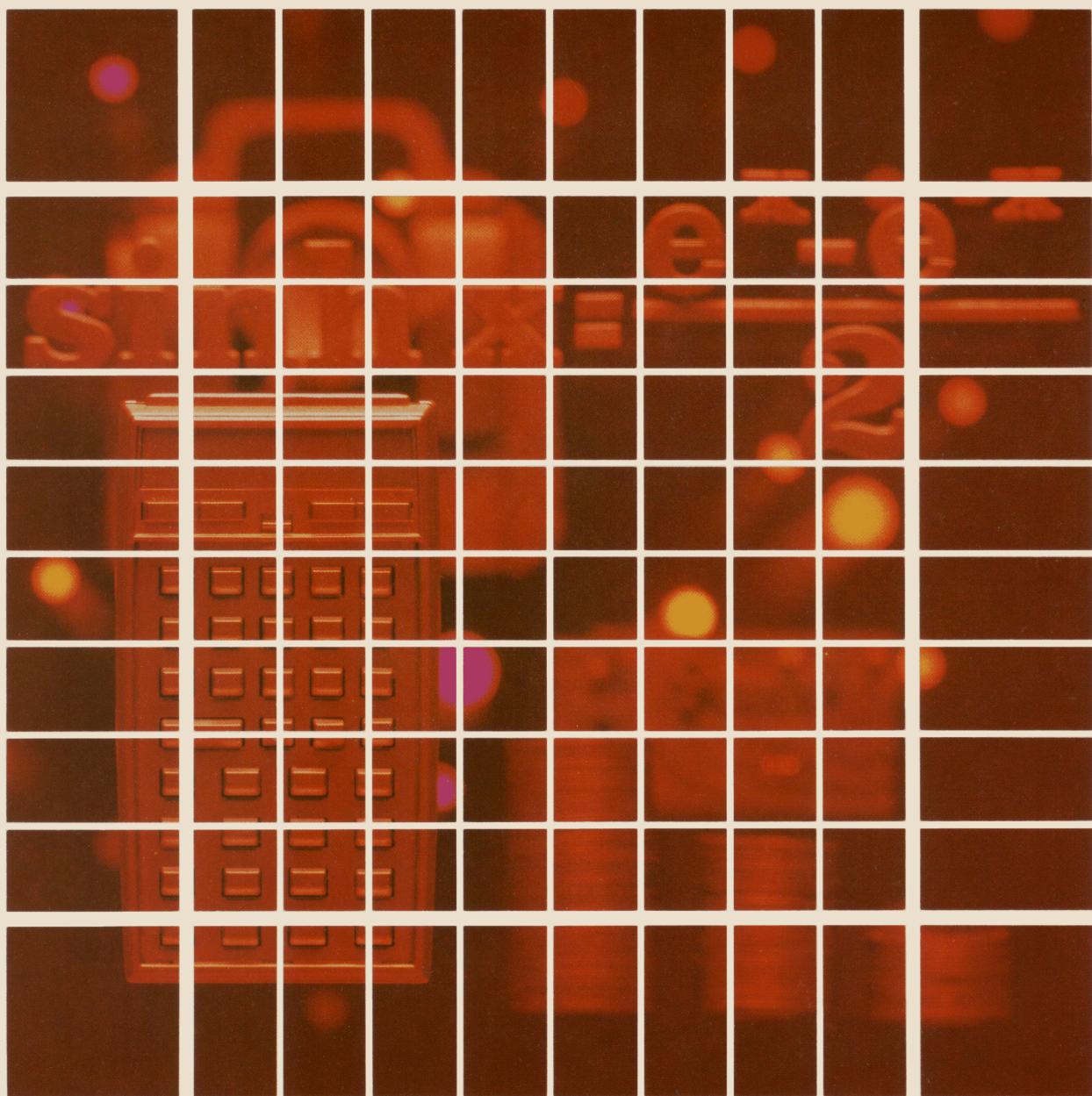


HEWLETT-PACKARD

HP-41C

USERS'  
LIBRARY SOLUTIONS  
Games



## **NOTICE**

The program material contained herein is supplied without representation or warranty of any kind. Hewlett-Packard Company therefore assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.

## INTRODUCTION

This HP-41C Solutions book was written to help you get the most from your calculator. The programs were chosen to provide useful calculations for many of the common problems encountered.

They will provide you with immediate capabilities in your everyday calculations and you will find them useful as guides to programming techniques for writing your own customized software. The comments on each program listing describe the approach used to reach the solution and help you follow the programmer's logic as you become an expert on your HP calculator.

## KEYING A PROGRAM INTO THE HP-41C

There are several things that you should keep in mind while you are keying in programs from the program listings provided in this book. The output from the HP 82143A printer provides a convenient way of listing and an easily understood method of keying in programs without showing every keystroke. This type of output is what appears in this handbook. Once you understand the procedure for keying programs from the printed listings, you will find this method simple and fast. Here is the procedure:

1. At the end of each program listing is a listing of status information required to properly execute that program. Included is the SIZE allocation required. Before you begin keying in the program, press **XEQ ALPHA SIZE ALPHA** and specify the allocation (three digits; e.g., 10 should be specified as 010).

Also included in the status information is the display format and status of flags important to the program. To ensure proper execution, check to see that the display status of the HP-41C is set as specified and check to see that all applicable flags are set or clear as specified.

2. Set the HP-41C to PRGM mode (press the **PRGM** key) and press **■ GTO • •** to prepare the calculator for the new program.
3. Begin keying in the program. Following is a list of hints that will help you when you key in your programs from the program listings in this handbook.
  - a. When you see " (quote marks) around a character or group of characters in the program listing, those characters are ALPHA. To key them in, simply press **ALPHA**, key in the characters, then press **ALPHA** again. So "SAMPLE" would be keyed in as **ALPHA "SAMPLE" ALPHA**.
  - b. The diamond in front of each LBL instruction is only a visual aid to help you locate labels in the program listings. When you key in a program, ignore the diamond.
  - c. The printer indication of divide sign is /. When you see / in the program listing, press **+**.
  - d. The printer indication of the multiply sign is ×. When you see × in the program listing, press **×**.
  - e. The † character in the program listing is an indication of the **APPEND** function. When you see †, press **■ APPEND** in ALPHA mode (press **■** and the K key).
  - f. All operations requiring register addresses accept those addresses in these forms:  
nn (a two-digit number)  
IND nn (INDIRECT: **■**, followed by a two-digit number)  
X, Y, Z, T, or L (a STACK address: **•** followed by X, Y, Z, T, or L)  
IND X, Y, Z, T or L (INDIRECT stack: **■ •** followed by X, Y, Z, T, or L)

Indirect addresses are specified by pressing **■** and then the indirect address. Stack addresses are specified by pressing **•** followed by X, Y, Z, T, or L. Indirect stack addresses are specified by pressing **■ •** and X, Y, Z, T, or L.

### Printer Listing

```
01 ♦LBL "SAM  
PLE"  
02 "THIS IS  
A"  
03 "†SAMPLE  
"  
04 AVIEW  
05 6  
06 ENTER↑  
07 -2  
08 /  
09 ABS  
10 STO IND  
L  
11 "R3="  
12 ARCL 03  
13 AVIEW  
14 RTN
```

### Keystrokes

```
■ LBL ALPHA SAMPLE ALPHA  
ALPHA THIS IS A ALPHA  
ALPHA ■ APPEND SAMPLE  
■ AVIEW ALPHA  
6  
ENTER↑  
2 CHS  
+  
XEQ ALPHA ABS ALPHA  
STO ■ • L  
ALPHA R3= ■ ARCL 03  
■ AVIEW  
ALPHA  
■ RTN
```

### Display

```
01 LBL †SAMPLE  
02 †THIS IS A  
03 † †SAMPLE  
04 AVIEW  
05 6  
06 ENTER ↑  
07 -2  
08 /  
09 ABS  
10 STO IND L  
11 †R3=  
12 ARCL 03  
13 AVIEW  
14 RTN
```

## TABLE OF CONTENTS

*1.	HUNT THE WUMPUS . . . . .	1
	Somewhere in 20 caves is a cave with a Wumpus, 2 with pits and 2 with super bats. Barring any accidents, find and shoot the Wumpus.	
**2.	3-D TIC TAC TOE . . . . .	8
	On a 4x4x4 board, play against the 41C. The outcome is by no means certain.	
*3.	ROBOT TRAP . . . . .	17
	Control the android so that the advancing robots stumble into their own force fields. If you are not successful they will stomp you.	
**4.	HEXAPAWN . . . . .	24
	A 3x3 board game where you can punish and thus teach the 41C to play a better (or worse) game.	
*5.	SCATTER . . . . .	33
	Find the hidden atoms in a box by probing and watching the reflections.	
6.	FLIP-FLOP . . . . .	41
	A puzzle. Try to change the pattern to one which is given.	
*7.	ORBITAL LANDER . . . . .	46
	Land your orbiting lander on the moon.	
8.	PLANET LANDER . . . . .	53
	Execute the vertical descent part of a landing on any planet you wish.	
*9.	WARI . . . . .	59
	With 12 pits and 48 counters play either another person or the 41C.	
10.	SIMON.....	66
	If you can remember the sequence, you win.	

\* Requires one memory module  
 \*\* Requires two memory modules

## HUNT THE WUMPUS

(Requires 1 Memory Module)

There are 20 caves each connected to 3 others. There are 6 occupied caves: 2 with super bats, 2 with pits, 1 with a wumpus, and 1 with a hunter (you). The object is to move from cave to cave, watching the warnings to see what is nearby, find the wumpus and shoot it with an arrow. You may only move or shoot into an adjacent cave.

The warnings are as follows:

"SMELL WUMPUS" -- The wumpus is in an adjacent cave.

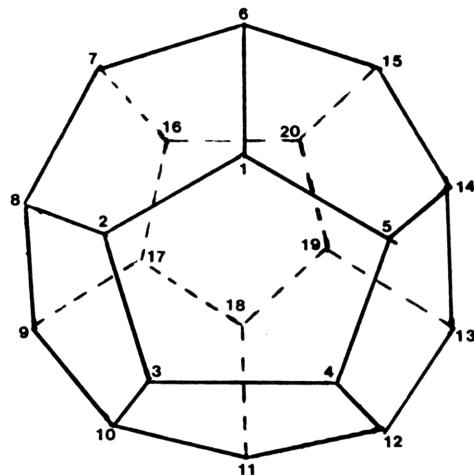
"FEEL A DRAFT" -- Pits are in adjacent cave(s).

"HEAR SQUEEKS" -- Super bats are in adjacent cave(s).

If you walk into the cave with the wumpus he will eat you. If you shoot an arrow and miss (by shooting into the wrong cave), you wake the wumpus up and he moves to one of his adjacent caves. If it's the cave you are in, he eats you. Since he has sucker feet he can walk into a cave with pits without falling. And, since he is too heavy for the bats, they cannot bother him. If you walk into a cave with pits, you fall and the game is ended. If you walk into a cave with super bats, they will carry you to some other cave randomly chosen. If the wumpus, pits, or more super bats are there, you will suffer their consequences. The bats then return to their original cave.

Each time you move you will first see the consequences (if any) of your move: "CHOMP" means you've been eaten by the wumpus, "YYIIIEEEE..." means you've fallen into a pit, and "SNATCH" followed by "MOVED TO ( )" means the super bats have grabbed you and moved you. After the consequences you see any warnings that are appropriate. Last, you will see the cave you are in followed by your adjacent caves.

The caves are arranged as a dodechedron with each vertex being a cave and each edge a tunnel.



**Example:****Keystrokes:**

[XEQ] [ALPHA] SIZE [ALPHA] 012  
[XEQ] [ALPHA] WUMPUS [ALPHA]  
.1232123 [R/S]  
[R/S]  
1 [A]  
[R/S]  
[R/S]  
[R/S]  
10 [A]  
[R/S]  
11 [A]  
[R/S]  
12 [B]  
18 [A]  
[R/S]  
19 [B]  
17 [A]  
[R/S]  
[R/S]  
9 [A]

(Note that you should have known pits were  
there from your pass through cave 10 and  
its warnings.)

**Display:**

SEED?  
HEAR SQUEAKS  
6-1,7,15  
SNATCH  
MOVED TO 3  
HEAR SQUEAKS  
3-4,2,10  
FEEL A DRAFT  
10-3,11,9  
SMELL WUMPUS  
11-18,12,10  
11-18,12,10  
SMELL WUMPUS  
18-11,19,17  
18-11,19,17  
SMELL WUMPUS  
FEEL A DRAFT  
17-9,18,16  
YYIIIEEEE...

# User Instructions

# Program Listings

01♦LBL "WUM PUS"		47 RCL 06 48 XEQ 21 49♦LBL 22	-----
02 0	Initialize	50 CLA 51 ARCL 01 52 "F- " 53 ARCL 07 54 "F, " 55 ARCL 08 56 "F, " 57 ARCL 09 58 AVIEW 59 RTN	Output current position
03 "SEED?"		60♦LBL 21 61 RCL 07 62 X<>Y 63 X=Y? 64 AVIEW 65 RCL 08 66 X<>Y 67 X=Y? 68 AVIEW 69 RCL 09 70 X=Y? 71 AVIEW 72 RTN	-----
04 PROMPT		73♦LBL A 74 XEQ 14 75 FS? 00 76 GTO 22 77 STO 01 78 GTO 09	-----
05 STO 00		79♦LBL B 80 XEQ 14	Move
06 SF 27		81 FS? 00 82 GTO 22	Shoot
07♦LBL E		83 RCL 02 84 X=Y? 85 GTO 11 86 XEQ 50 87 3 88 XEQ 99 89 7 90 +	Hit
08 FIX 0		91 RCL IND X	
09 CF 29		92 STO 02 93 RCL 01 94 XEQ 50 95♦LBL 09	New Wumpus position
10 1.006		96 2.006	-----
11 STO 10		97 STO 10	
12♦LBL 13			
13 RCL 10			
14 INT			
15 1			
16 -			
17 STO 11			
18 20			
19 XEQ 99	Pick a cave		
20 1			
21 +			
22♦LBL 12			
23 RCL IND			
11	Make sure it's not used		
24 X<>Y			
25 X=Y?			
26 GTO 13			
27 DSE 11			
28 GTO 12			
29 STO IND			
10			
30 ISG 10			
31 GTO 13			
32♦LBL C			
33♦LBL 20			
34 RCL 01	Check surroundings		
35 XEQ 50			
36 "SMELL W			
UMPUS"			
37 RCL 02			
38 XEQ 21			
39 "FEEL A			
DRAFT"			
40 RCL 03			
41 XEQ 21			
42 RCL 04			
43 XEQ 21			
44 "HEAR SQ			
UEEKS"			
45 RCL 05			
46 XEQ 21			

# Program Listings

98 RCL 01		145 RCL 09	
99♦LBL 08		146 X=Y?	
100 RCL IND	Check current position for dangers	147 RTN	
10		148 "ILLEGAL CAVE"	
101 X<>Y		149 AVIEW	
102 X=Y?		150 SF 00	
103 GTO IND		151 RTN	
10		152♦LBL 99	
104 ISG 10		153 RCL 00	Random number generator
105 GTO 08		154 9821	
106 GTO 20		155 *	
107♦LBL 02	-----	156 .21137	
108 "CHOMP"	Eaten by Wumpus	157 +	
109 AVIEW		158 FRC	
110 RTN	-----	159 STO 00	
111♦LBL 03		160 *	
112♦LBL 04	Fell into pit	161 INT	
113 "YYIIIEEE		162 RTN	
EE..."		163♦LBL 50	
114 AVIEW		164 CF 00	
115 RTN	-----	165 5	
116♦LBL 05		166 X<>Y	
117♦LBL 06		167 STO 10	
118 "SNATCH"	Super bat move	168 X<=Y?	
119 AVIEW		169 GTO 10	
120 20		170 15	Determine which ring the cave is in
121 XEQ 99		171 X<>Y	
122 1		172 X<=Y?	
123 +		173 GTO 11	
124 STO 01		174 2	
125 "MOVED T		175 *	
O "		176 25	Find adjacent caves for outside ring
126 ARCL 01		177 -	
127 AVIEW		178 STO 07	
128 PSE	-----	179 21	
129 GTO 09		180 RCL 10	
130♦LBL 11		181 1	
131 "GOT HIM"	Shoot Wumpus	182 +	
"		183 16	
132 BEEP		184 RDN	
133 AVIEW		185 X=Y?	
134 RTN	-----	186 RT	
135♦LBL 14		187 STO 08	
136 CF 00		188 15	
137 RCL 07	Illegal cave test	189 RCL 10	
138 X=Y?		190 1	
139 RTN		191 -	
140 X<>Y		192 X=Y?	
141 RCL 08		193 20	
142 X=Y?		194 X=0?	
143 RTN		195 RDN	
144 X<>Y			

# Program Listings

196 STO 09		248 RT
197 RTN		249 STO 09
198♦LBL 10	-----	250 RTN
199 1		251 .END.
200 +	Find adjacent caves for inside ring	
201 X>Y?		
202 1		
203 STO 07		
204 RCL 10		
205 1		60
206 -		
207 X=0?		
208 5		
209 STO 08		
210 RCL 10		
211 2		
212 *		
213 4		
214 +		
215 STO 09		
216 RTN	-----	70
217♦LBL 11		
218 2		
219 /	Find adjacent caves for middle ring	
220 FRC		
221 X=0?		
222 SF 00		
223 25		
224 RCL 10		
225 FS? 00		
226 4		80
227 FS? 00		
228 CHS		
229 CF 00		
230 +		
231 2		
232 /		
233 STO 07		
234 16		
235 RCL 10		
236 1		
237 +		90
238 X=Y?		
239 6		
240 STO 08		
241 5		
242 RCL 10		
243 1		
244 -		
245 15		
246 RDN		
247 X=Y?		00

# REGISTERS, STATUS, FLAGS, ASSIGNMENTS<sup>7</sup>

DATA REGISTERS				STATUS				
				SIZE	012	TOT. REG.	78	USER MODE
#	NAME	INIT	S/C	SET	INDICATES	CLEAR	INDICATES	
00	Seed	50						
	Hunter							
	Wumpus							
	Pits							
	Pits							
05	S. bats	55						
	S. bats							
	Adj. cave			00	Used		Used	
	Adj. cave							
	Adj. cave							
10	Used	60						
	Used							
	Used							
15		65						
20		70						
25		75						
30		80						
35		85						
				ASSIGNMENTS				
				FUNCTION	KEY	FUNCTION	KEY	
40		90						
45		95						

## 3-D TIC TAC TOE

(Requires 2 Memory Modules)

This program pits the HP-41C against a human opponent in a game of 3-D Tic Tac Toe. The rules of this game are simple:

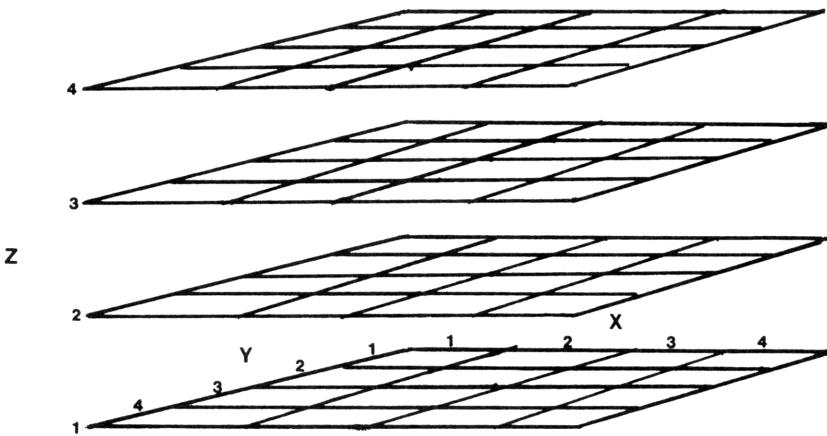
- 1) The board consists of 4 levels, each of which is 4 rows deep and 4 columns across, making a total of 64 squares on a 3 dimensional board.
- 2) Two players move alternately by placing a black or white marker on a square (or making an X or a 0 on a paper layout of the board). Once a move is made, the piece is never moved or removed. In this game, the human always goes first.
- 3) A player wins by placing four markers in a straight line. The line can lie in more than one level, and diagonals are perfectly legitimate wins.

In short, the game is played just like regular Tic Tac Toe, except that the board has one additional dimension, and is one square bigger in all dimensions. Unlike regular Tic Tac Toe, there is no known winning strategy for the 3-D version. It is a much more complex game which can require considerable skill in a player, allowing for very complicated strategies.

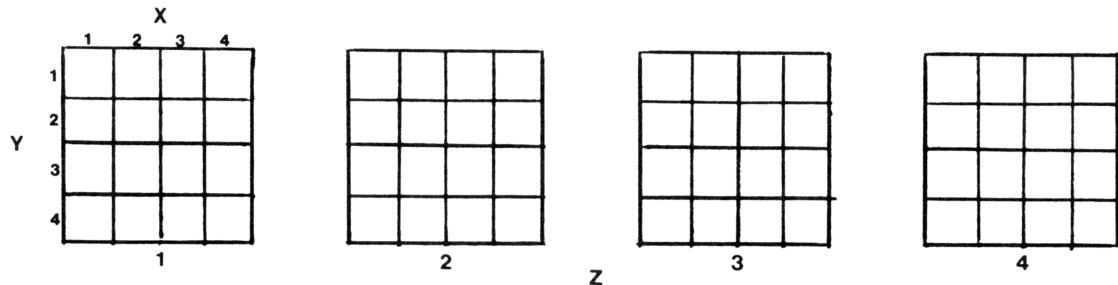
The 41C plays and remembers the game by dividing the board into its 16 component rows and storing an entire row in one register. The registers  $R_0$  through  $R_{15}$  are reserved for the game board.

Each square on the board can be characterized by its level= $z$ , its row= $y$  and its column= $x$ .  $x, y$ , and  $z$  can have values from 1 through 4. When entering moves, make sure they are 3 digit numbers. All three digits must be between 1 and 4 inclusive. Entering a move outside this range may cause the program to make erroneous entries in the board.

The boards look like:



or



Example:

Keystrokes:

```
[XEQ] [ALPHA] SIZE [ALPHA] 026
[XEQ] [ALPHA] 3DTTT [ALPHA]
242 [R/S]
414 [R/S]
123 [R/S]
441 [R/S]
141 [R/S]
214 [R/S]
424 [R/S]
111 [R/S]
```

Display:

```
READY
MY MOVE: 322
MY MOVE: 134
MY MOVE: 234
MY MOVE: 423
MY MOVE: 232
MY MOVE: 434
MY MOVE: 114
MY MOVE: 434
334, I WIN
```

The boards look like:

	1	2	X	3	4
1	X				
2					
3					
4	X			X	

1

			0
	0		
	X		

2

			0
X			

3

	0	X		X
				X
0	0	0	0	0

4

Z



# Program Listings

11

01♦LBL "3DT TT" 02 FIX 0 03 CLRG 04 CF 00 05 CF 01 06 CF 02 07 CF 03 08 CF 29 09 "READY" 10 AVIEW 11♦LBL A 12 STOP 13 1 E3 14 / 15 STO 19 16 0 17 STO 16 18 2 19 STO 17 20 RDN 21 RDN 22 XEQ 10 23 STO 25 24 RCL 20 25 RCL IND  25 26 * 27 INT 28 1 E2 29 / 30 FRC 31 "ILLEGAL MOVE" 32 X#0? 33 AVIEW 34 X#0? 35 GTO A 36 5 37 RCL 20 38 / 39 1 40 + 41 ST+ IND 25 42 RCL 18 43 X=0? 44 GTO 01 45 SF 00 46 XEQ 04 47 CF 02	Initialize ----- Input players move ----- Illegal move ----- Store move ----- Test move	48 CF 03 49 CF 00 50 .06 51 RCL 18 52 FRC 53 STO 18 54 X>Y? 55 GTO 02 56 X=Y? 57 GTO 03 58♦LBL 01 59 1 E-2 60 ST+ 18 61 SF 02 62 GTO 03 63♦LBL 02 64 XEQ 10 65 XEQ 04 66 FS?C 02 67 GTO 03 68 RCL 19 69 XEQ 10 70♦LBL 03 71 RCL 16 72 INT 73 1 74 - 75 3 76 / 77 21 78 + 79 ENTER↑ 80 FRC 81 X=0? 82 CF 01 83 .5 84 X>Y? 85 SF 00 86 GTO IND Z 87♦LBL 10 88 10 89 * 90 INT 91 LASTX 92 FRC 93 10 94 * 95 INT 96 LASTX 97 FRC 98 .5	----- Increment round count ----- Test machine move ----- Jump to proper test routine ----- Parse move
---	---	---	---

# Program Listings

99 -		151♦LBL 02	
100 CHS		152 1	Testing cycle
101 20		153 ST+ 18	
102 *		154 1 E2	
103 10↑X		155 ENTER↑	
104 STO 20		156 ENTER↑	
105 RDN		157 RCL IND	
106 1		25	
107 -		158 X<> 25	
108 STO 21		159 RCL 24	
109 X<>Y		160 +	
110 1		161 X<> 25	
111 -		162 FS? 01	
112 4		163 *	
113 *		164 RCL IND	
114 STO 22		25	
115 +		165 +	
116 STO 23		166 X<> 25	
117 RTN		167 RCL 24	
118♦LBL 04	-----	168 +	
119 1	Initialize test	169 X<> 25	
120 RCL 22	controls	170 FS? 01	
121 XEQ 01		171 *	
122 4		172 RCL IND	
123 RCL 21		25	
124 XEQ 01		173 +	
125 5		174 X<> 25	
126 ENTER↑		175 RCL 24	
127 0		176 +	
128 XEQ 01		177 X<> 25	
129 3		178 FS? 01	
130 ENTER↑		179 *	
131 XEQ 01		180 RCL IND	
132 0		25	
133 STO 24		181 +	
134 RCL 23		182 FS? 01	
135 STO 25		183 GTO 01	
136 GTO 02	-----	184 R↑	
137♦LBL 01		185 RCL 20	
138 CF 01		186 /	
139 STO 25	Set test points	187 /	
140 RDN		188♦LBL 01	-----
141 STO 24		189 FRC	Analyze sum
142 XEQ 02		190 R↑	
143 SF 01		191 *	
144 RCL 24		192 INT	
145 CHS		193 4	
146 STO 24		194 X<>Y	
147 XEQ 02		195 FS? 00	
148 RCL 24		196 GTO 01	
149 CHS		197 X>Y?	
150 STO 24		198 RTN	

# Program Listings

199 GTO 02	-----	250 FS? 00	
200♦LBL 01		251 X<>Y	
201 5	Analyze player	252 FS? 01	
202 /	sum	253 STO 20	
203 "YOU WIN		254 RCL 20	
"		255♦LBL 05	-----
204 X=Y?		256 XEQ 19	
205 PROMPT	Player wins	257 X<> 25	Find move
206 FRC		258 RCL 24	
207 X≠0?		259 +	
208 RTN		260 X<> 25	
209 LASTX		261 1 E-2	
210♦LBL 02	-----	262 FS? 00	
211 RCL 17	Test tactical	263 1/X	
212 X>Y?	status	264 RCL 20	
213 RTN		265 FS? 01	
214 X<>Y		266 *	
215 SF 02		267 GTO 05	-----
216 STO 17		268♦LBL 20	
217 FC? 00		269 16	
218 SF 03		270 STO 20	Strategic play
219 RCL 18		271 4	
220 STO 16		272 STO 25	
221 RTN		273 1	
222♦LBL 21	-----	274 STO 24	
223 1		275 XEQ 07	
224 RCL 22		276 2	
225 GTO 01		277 XEQ 07	
226♦LBL 22		278 3	
227 4		279 XEQ 07	
228 RCL 21		280 0	
229 GTO 01		281 XEQ 08	
230♦LBL 23		282 RCL 22	
231 5		283 4	
232 ENTER↑		284 STO 24	
233 0		285 *	
234 GTO 01		286 STO 23	
235♦LBL 24		287 16	
236 3		288 STO 20	
237 ENTER↑		289 1	
238 GTO 01		290 XEQ 08	
239♦LBL 25		291 2	
240 SF 01		292 XEQ 08	
241 0		293 3	
242 RCL 23		294 XEQ 08	
243♦LBL 01	Tactical play	295 0	
244 STO 25		296 XEQ 08	
245 RDN		297 RCL 22	
246 STO 24		298 RCL 23	
247 1 E2		299 +	
248 ENTER↑		300 STO 25	-----
249 1 E8			

# Program Listings

301♦LBL 09		348 INT	
302 RCL IND	Find move	349 RCL 20	
25		350 X<=Y?	
303 RCL 24		351 RTN	
304 X<=Y?		352 RDN	
305 GTO 01		353 STO 20	
306 RCL 25		354 RDN	
307 2		355 STO 22	
308 /		356 RTN	
309 FRC		357♦LBL 02	
310 X=0?		358 1 E2	
311 GTO 02		359 XEQ 19	Find empty space
312 GTO 03		360 1 E4	
313♦LBL 01		361 XEQ 19	
314 RCL 25	Reset reg. #	362 1 E8	
315 +		363 XEQ 19	
316 16		364♦LBL 03	
317 X>Y?		365 1 E6	
318 CLX		366 XEQ 19	
319 -		367 1 E4	
320 STO 25		368 XEQ 19	
321 GTO 09		369 1 E2	
322♦LBL 08		370 XEQ 19	
323 STO 25		371 1 E8	
324♦LBL 07	Find total moves	372♦LBL 19	
325 RCL IND		373 STO 20	
25		374 RCL IND	
326 X<> 25		25	
327 RCL 24		375 *	
328 +		376 INT	
329 X<> 25		377 1 E2	
330 RCL IND		378 /	
25		379 FRC	
331 +		380 X≠0?	
332 X<> 25		381 RTN	
333 RCL 24		382 RCL 20	
334 +		383 1/X	
335 X<> 25		384 ST+ IND	
336 RCL IND		25	
25		385 LOG	
337 +		386 2	
338 X<> 25		387 /	
339 RCL 24		388 5	
340 +		389 +	
341 X<> 25		390 RCL 25	
342 RCL IND		391 4	
25		392 /	
343 +		393 INT	
344 X<> 25		394 LASTX	
345 RCL 24		395 FRC	
346 +		396 4	
347 X<> 25			

# Program Listings

397 *		51	
398 1			
399 ST+ IND			
25			
400 +			
401 X<>Y			
402 1			
403 +			
404 10	Place in XYZ form		
405 *		60	
406 +			
407 10			
408 *			
409 +			
410 CLA			
411 ARCL X			
412 1 E3			
413 /			
414 FS?C 02			
415 GTO 01			
416 STO 18		70	
417 RCL 17			
418 3			
419 X>Y?			
420 GTO 01			
421 FC? 03			
422 GTO 01			
423 "F, I WI N"			
424 AVIEW	41C wins		
425 GTO A		80	
426♦LBL 01			
427 ASTO X			
428 "MY MOVE			
:			
429 ARCL X			
430 AVIEW			
431 GTO A	Get next player move		
432 .END.			
40		90	
50		00	

# <sup>16</sup> REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS			STATUS			
00	Board	50		SIZE	026	TOT. REG. 133
				ENG	FIX	SCI
				DEG	RAD	GRAD
05		55		FLAGS		
				#	INIT S/C	SET INDICATES
				00		Used
				01		Used
				02		Used
10		60		03		Used
15		65				
	Test #					
	Tactic status					
	mach. move					
	player move					
20	Used	70				
	Y-1					
	Z-1					
	Register #					
	Position Inc.					
25	Used	75				
	:					
30		80				
35		85		ASSIGNMENTS		
				FUNCTION	KEY	FUNCTION
40		90				
45		95				

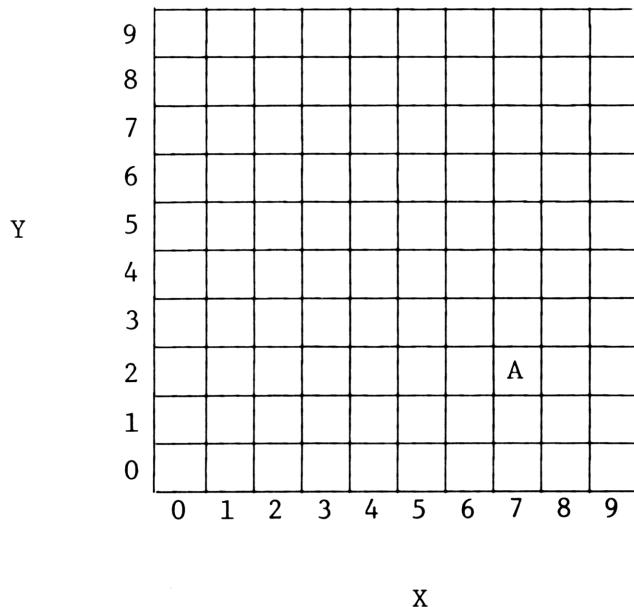
## ROBOT TRAP

(Requires at least 1 Memory Module)

You move your android to any adjacent square on a 10 x 10 playing board studded with destructive force fields and up to 49 enemy robots in such a way as to lure the robots into their own electronic booby traps and save the android. The robots will always close on the android, moving to the square adjacent to their present position which is nearer to the row and column position of the android, and will team up to destroy him. The android, like the robots, is destroyed by moving into a force field, and the android is also destroyed by colliding with a robot. If robots collide all but one involved are destroyed. You choose the initial number of robots. The number of force fields is equal to the initial number of robots plus one. Even a few robots can be challenging and the more robots the more difficult. All initial positions are randomly generated.

If you move the android into a force field you will see "ZAP" then "TOO BAD". If you move the android into a robot you will see "STOMP" then "TOO BAD". If a robot stumbles into a force field you will see "STUMBLE". And, if robots hit, you will see "BUMP". Finally, if all robots are destroyed, you will see "YOU WIN".

The board is set up as below:



Board positions are denoted as x.y . The android shown is at 7,2 . To move the android use the digits keys to specify directions as follows:

- |                    |                  |
|--------------------|------------------|
| 1 - down and left  | 6 - right        |
| 2 - down           | 7 - up and left  |
| 3 - down and right | 8 - up           |
| 4 - left           | 9 - up and right |
| 5 - no movement    |                  |

Note: Do not move off the board. Execution times at all points in the program increase with the initial number of robots.

Example: Try to destroy robots.

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 014

[XEQ] [ALPHA] RT [ALPHA]

.12569 [R/S]

3 [R/S]

[R/S]

[R/S]

[R/S]

[R/S]

[R/S]

[R/S]

[R/S]

3 [R/S]

[R/S]

[R/S]

4 [R/S]

[R/S]

[R/S]

.

.

.

Display:

SEED?

NO. OF ROBOTS

F.F. AT 7,8

F.F. AT 7,1

F.F. AT 3,6

F.F. AT 3,3

ROBOT AT 6,8

ROBOT AT 0,6

ROBOT AT 7,9

ANDROID: 6,1

STUMBLE

ROBOT AT 7,7

ROBOT AT 1,5

ANDROID: 7,0

ROBOT AT 6,6

ROBOT AT 2,4

ANDROID: 6,0

.

.

.

# User Instructions

# Program Listings

<pre> 01♦LBL "RT" 02 SF 27 03 "SEED?" 04 PROMPT 05 STO 00 06♦LBL E 07 "NO. OF ROBOTS" 08 PROMPT 09 STO 03 10 2 11 * 12 1 13 + 14 6 15 + 16 "SET SIZ E" 17 FIX 0 18 CF 29 19 ARCL X 20 SF 25 21 1 22 - 23 STO IND X 24 FC?C 25 25 PROMPT 26 RCL 03 27 1 E3 28 / 29 ST+ 03 30 6.005 31 + 32 STO 02 33 RCL 03 34 + 35 .001 36 + 37 STO 01 38 FRC 39 6 40 + 41 STO 03 42 5.004 43 STO 04 44 100 45 XEQ 99 46 10 47 / 48 STO 05 </pre>	<p>Initialize</p> <p>Calculate SIZE</p> <p>See if reg. exists</p>	<pre> 49♦LBL 00 50 RCL 04 51 100 52 XEQ 99 53 10 54 / 55♦LBL 11 56 RCL IND Y 57 X=Y? 58 GTO 00 59 RDN 60 DSE Y 61 GTO 11 62 STO IND 03 63 1 64 ST+ 04 65 ISG 03 66 GTO 00 67 FIX 1 68 CF 28 69 RCL 01 70 "F. F. " 71 ASTO 03 72 XEQ 10 73♦LBL 98 74 CF 28 75 RCL 02 76 "ROBOT " 77 ASTO 03 78 CF 05 79 XEQ 10 80 "ANDROID " 81 ARCL 05 82 AVIEW 83 SF 28 84 FS? 05 85 GTO 97 86 PSE 87 "ANDROID SAFE" 88 AVIEW 89 TONE 9 90 TONE 8 91 TONE 9 92 TONE 7 93 TONE 9 94 RTN 95♦LBL 10 96 STO 04 </pre>	<p>Get positions</p> <p>Make sure no two are the same</p> <p>Output F.F's once</p> <p>Output Robots</p> <p>Safe time</p> <p>-----</p>
---	---	--	---

# Program Listings

97♦LBL 16		147 XEQ 96	
98 RCL IND	Output loop	148 FS? 05	
04		149 GTO 95	Android blunder
99 X<0?		150 RCL 02	
100 GTO 10		151 STO 03	
101 SF 05		152♦LBL 14	
102 CLA		153 RCL 05	Move robots
103 ARCL 03		154 INT	
104 "HAT "		155 RCL IND	
105 ARCL X		03	Recall robot
106 AVIEW		156 X<0?	
107 STOP		157 GTO 10	
108♦LBL 10		158 INT	
109 ISG 04		159 -	
110 GTO 16		160 X≠0?	
111 RTN		161 SIGN	
112♦LBL 97		162 RCL 05	
113 CF 22		163 FRC	
114 STOP		164 RCL IND	
115 FC? 22		03	
116 GTO 98		165 FRC	
117 GTO IND		166 -	
X		167 X≠0?	
118♦LBL 01	Change digit	168 SIGN	
119 -.1	into ΔXY	169 10	
120 GTO 10		170 /	
121♦LBL 02		171 +	
122 -.1		172 ST+ IND	Update robot
123 GTO 10		03	
124♦LBL 03		173♦LBL 10	
125 .9		174 ISG 03	
126 GTO 10		175 GTO 14	
127♦LBL 04		176 RCL 02	
128 -1		177 STO 03	
129 GTO 10		178♦LBL 15	
130♦LBL 05		179 -1	
131 0		180 X<> IND	
132 GTO 10		03	
133♦LBL 06		181 X<0?	
134 1		182 GTO 10	
135 GTO 10		183 XEQ 96	
136♦LBL 07		184 FC? 05	
137 -.9		185 GTO 10	
138 GTO 10		186 -1	
139♦LBL 08		187 "BUMP"	
140 .1		188 FS? 06	
141 GTO 10		189 "STUMBLE	
142♦LBL 09		"	
143 1.1		190 AVIEW	
144♦LBL 10	Update android	191 TONE 9	
145 ST+ 05	position	192 CLD	
146 RCL 05		193♦LBL 10	

# Program Listings

194 STO IND		242 *
03		243 INT
195 ISG 03		244 RTN
196 GTO 15		245 .END.
197 RCL 05	Robot get android?	
198 XEQ 96		
199 FC? 05		
200 GTO 98		
201♦LBL 95		
202 "STOMP"	Android loses	60
203 FS? 06		
204 "ZAP"		
205 AVIEW		
206 TONE 0		
207 "TOO BAD		
"		
208 AVIEW		
209 RTN		
210♦LBL 96		
211 SF 05	Android or robot	70
212 SF 06	blunder routine	
213 RCL 01		
214 X<>Y		
215♦LBL 12		
216 RCL IND		
Y		
217 X=Y?		
218 RTN		
219 RDN		
220 ISG Y		
221 GTO 12		80
222 CF 06		
223 RCL 02		
224 X<>Y		
225♦LBL 13		
226 RCL IND		
Y		
227 X=Y?		
228 RTN		
229 RDN		
230 ISG Y		
231 GTO 13		90
232 CF 05		
233 RTN		
234♦LBL 99		
235 RCL 00	Randon number	
236 9821	generator	
237 *		
238 .211327		
239 +		
240 FRC		
241 STO 00		00

# REGISTERS, STATUS, FLAGS, ASSIGNMENTS<sup>23</sup>

DATA REGISTERS				STATUS					
				SIZE	6+	TOT. REG.	76+	USER MODE	
				ENG		SCI		ON OFF	
				DEG		RAD		GRAD	
00	Seed	50							
	Used								
	Used								
	Used								
	Used								
05	Used	55							
	Android								
	Robot <sub>1</sub>			INIT		SET INDICATES		CLEAR INDICATES	
	⋮			#	S/C				
10	Robot <sub>n</sub>	60		05		Used			
	FF <sub>1</sub>			06		Robot		F.F.	
	⋮								
	FF <sub>n+1</sub>								
15		65							
20		70							
25		75							
30		80							
35		85							
				ASSIGNMENTS					
				FUNCTION	KEY	FUNCTION	KEY		
40		90							
45		95							

## HEXAPAWN

(Requires 2 Memory Modules)

Hexapawn is a game which is programmed to learn from its mistakes. The game is played with chess pawns on a 3x3 board. Pawns may advance one square at a time or capture the opponent's pawns by moving diagonally one square. The game starts with the pawns positioned as follows:

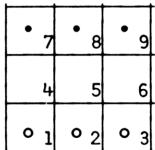
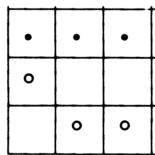
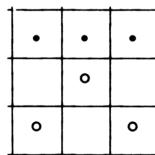


Figure 1. Starting position of pawns.

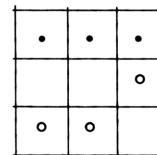
Note the resemblance between the 41C digit keys and the board numbering. The three allowed opening moves for the first player (in this example, white) are 1 to 4 (keyed as 1.4), 2 to 5 (2.5), and 3 to 6 (3.6).



1 to 4



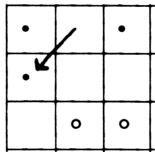
2 to 5



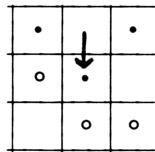
3 to 6

Figure 2. Opening moves

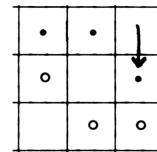
Black's three possible responses to white's 1.4 move are 8 to 4, 8 to 5, and 9 to 6.



8 to 4



8 to 5



9 to 6

Figure 3. Black's responses to white's 1.4 move

Black can move diagonally and capture white (8 to 4), or he can move either pawns 8 or 9 straight ahead one square. The black pawn at 7 is blocked. Note that the only way a pawn can move to an open square is straight ahead. Also the only way a pawn can capture is by moving diagonally.

The game is won by advancing a pawn to the third row, capturing all of the opponent's pawns, or creating a position in which the opponent cannot move.

Moves are made by keying in the board position of the pawn to be moved, a decimal point, then the board position the pawn is to be moved to. The 41C does not

check for illegal moves; therefore, you are on your honor not to cheat. The 41C selects its move at random, but if it is then punished, it remembers not to make that move in that situation. Thus, if the machine makes a poor move and is punished, it will not repeat the mistake.\* Also, if the mirror image game is played, it will not make the mirror image of the poor move. If a point is reached in a game where all possible moves for a certain board configuration have received previous punishment, "NO MOVE" and "YOU WIN" is displayed, just as if there really were no move. If you cannot move, you can if you wish be a good sport and tell the 41C by keying 0 for your move. It will respond with "I WIN". If chess pawns are not available for visualization, different colored coins work well.

\*Similarly, you can punish good moves to make it play a losing game.

Example 1: Let the 41C go first.

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 014

[XEQ] [ALPHA] MACHINE [ALPHA]

.1111111111 [R/S]

1.5 [R/S]

(A bad move; therefore, punish)

[E]

5.8 [R/S]

Display:

SEED?

8 to 5

7 to 4

AAAI...  
AAIIII...

YOU WIN

Example 2: Start a new game with the 41C remembering its punishment.

Keystrokes:

[A]

1.5 [R/S]

3.5 [R/S]

0 [R/S]

Display:

8 to 5

7 to 5

9 to 5

I WIN

# User Instructions

# Program Listings

01♦LBL "MAC HINE" 02 XEQ 01 03 8388607 04 STO 01 05 3139583 06 STO 02 07 34314 08 STO 03 09 SF 05 10 GTO A 11♦LBL "HUM AN" 12 XEQ 01 13 16777215 14 STO 01 15 16756735 16 STO 02 17 524413 18 STO 03 19 CF 05 20♦LBL A 21 SF 09 22 SF 08 23 9503 24 STO 10 25 .8596 26 STO 06 27 1 28 FS? 05 29 GTO 20 30 "READY" 31 AVIEW 32 GTO 30 33♦LBL 01 34 SF 27 35 CF 07 36 CLRG 37 "SEED?" 38 RCL 00 39 PROMPT 40 STO 00 41 RTN 42♦LBL 30 43 STOP 44 "I WIN" 45 X=0? 46 PROMPT 47 FC?C 08 48 GTO 00 49 CF 07	Machine first  ----- Human first  ----- Start game  ----- Initialize  ----- Get a human move	50 3.6 51 FS? 05 52 CHS 53 X=Y? 54 SF 07 55 RDN 56 1.4 57 FC? 05 58 CHS 59 X=Y? 60 SF 07 61 RDN 62 1.5 63 X=Y? 64 SF 07 65 RDN 66♦LBL 00 67 CF 09 68 STO 13 69 FRC 70 .7 71 X<=Y? 72 GTO 26 73 RCL 13 74 XEQ 50 75 INT 76 STO 11 77 LASTX 78 FRC 79 10 80 * 81 STO 12 82 CF 06 83 XEQ 21 84 .9 85 STO 13 86 FS? 05 87 GTO 22 88 3185.848 596 89 XEQ 23 90 7397.747 5 91 XEQ 23 92 1316.417 596 93 XEQ 23 94 1142.845 396 95 XEQ 23 96 2531.759 596	From  To  -----  Human move first boards and responses  (Watch for extra digits on a second line)
---	--	---	---

# Program Listings

97 XEQ 23		140 7341.748	
98 1023.848		586	
586		141 XEQ 23	
99 XEQ 23		142 7449.747	
100 6758.515		5	
286		143 XEQ 23	
101 XEQ 23		144 3237.848	
102 7163.958		562	
6		145 XEQ 23	
103 XEQ 23		146 8957.745	
104 2720.414		1	
2		147 XEQ 23	
105 XEQ 23		148 8849.959	
106 1131.847		6	
5		149 SF 09	
107 XEQ 23		150 XEQ 23	
108 818.9596		151 CF 09	
109 XEQ 23		152 6.9	
110 6650.959		153 STO 13	
6		154 8849.747	
111 XEQ 23		5	
112 992.96		155 XEQ 23	
113 XEQ 23		156 6687.747	
114 677.4152		5	
115 XEQ 23		157 XEQ 23	
116 369.75		158 855.7475	
117 XEQ 23		159 XEQ 23	
118 600.4186		160 1194.845	
119 XEQ 23		253	
120 384.8463		161 XEQ 23	
121 XEQ 23		162 2583.756	
122 693.4152		263	
123 XEQ 23		163 XEQ 23	
124 461.5263		164 6702.63	
125 XEQ 23		165 XEQ 23	
126 569.4195		166 1260.41	
96		167 XEQ 23	
127 XEQ 23		168 1368.754	
128 411.8452		1	
129 XEQ 23		169 XEQ 23	
130 195.5286		170 6887.959	
131 XEQ 23		6	
132 585.4175		171 XEQ 23	
133 XEQ 23		172 2783.414	
134 137.9563		295	
135 XEQ 23		173 XEQ 23	
136 GTO 25		174 6995.515	
137♦LBL 22	-----	2	
138 9503.859	Machine move first boards and responses	175 XEQ 23	
6		176 1179.525	
139 XEQ 23		3	

# Program Listings

177 XEQ 23	226 RCL 04	
178 2286.747	227 1	Move loop
5	228 +	
179 XEQ 23	229 X>Y?	
180 2270.959	230 1	
6	231 STO 04	
181 XEQ 23	232 X<> 13	
182 2594.96	233 RCL IND	
183 XEQ 23	13	
184 621.4163	234 X<>Y	
185 XEQ 23	235 X<> 13	
186 432.52	236 RDN	
187 XEQ 23	237 RCL 05	
188 GTO 25	238 /	
189♦LBL 23	239 FRC	
190 ISG 13	240 .5	
191 INT	241 X<=Y?	
192 RCL 10	Check to see if correct board	
193 X≠Y?	242 GTO 04	
194 RTN	243 DSE 13	
195 LASTX	244 GTO 02	
196 FRC	245 RCL 07	
197 STO 06	246 STO 05	
198 RCL 13	247 STO 04	
199 INT	248 FS? 08	
200♦LBL 20	249 CF 09	
201 STO 08	250 FS? 09	
202 RCL 04	251 RTN	
203 STO 09	252♦LBL 25	
204 RCL 05	253 "NO MOVE	No move possible
205 STO 07	"	
206 2	254 AVIEW	
207 RCL 08	255 PSE	
208 Y↑X	256♦LBL 26	
209 STO 05	257 "YOU WIN	
210 3	"	
211 STO 13	258 AVIEW	
212 RCL 00	259 STOP	
213 9821	260♦LBL 04	
214 *	261 RCL 04	
215 .211327	262 1	Convert machines move to board notation
216 +	263 X≠Y?	
217 FRC	264 CF 08	
218 STO 00	265 -	
219 *	266 2	
220 1	267 *	
221 +	268 10↑X	
222 INT	269 RCL 06	
223 STO 04	270 *	
224♦LBL 02	271 FRC	
225 3	272 10	
	273 *	

# Program Listings

274 INT		324 -	
275 STO 11		325 RTN	
276 LASTX		326♦LBL 05	
277 FRC		327 5	
278 10		328 RTN	
279 *		329♦LBL 06	
280 INT		330 10.4	
281 STO 12		331 RTN	
282 SF 06		332♦LBL 07	
283 XEQ 21		333 17	
284 3		334 RTN	
285 RCL 12		335♦LBL 21	
286 X>Y?		336 RCL 10	
287 GTO 00	41C win	337 3	Update board
288 AVIEW		338 ENTER↑	
289 BEEP		339 10	
290♦LBL 00	-----	340 ENTER↑	
291 RCL 11		341 RCL 12	
292 RCL 12	41C move display and board update	342 -	
293 10		343 Y↑X	
294 /		344 /	
295 +		345 FRC	
296 XEQ 50		346 3	
297 FIX 0		347 *	
298 CF 29		348 INT	
299 CLA		349 CHS	
300 INT		350 1	
301 ARCL X		351 FS? 06	
302 "F TO "		352 ST+ X	
303 LASTX		353 +	
304 FRC		354 3	
305 10		355 ENTER↑	
306 *		356 9	
307 ARCL X		357 ENTER↑	
308 AVIEW		358 RCL 12	
309 GTO 30	-----	359 -	
310♦LBL 50		360 Y↑X	
311 FC? 07		361 *	
312 RTN	Get mirror image move	362 3	
313 STO 06		363 ENTER↑	
314 INT		364 9	
315 1		365 RCL 11	
316 -		366 -	
317 3		367 Y↑X	
318 /		368 1	
319 INT		369 FS? 06	
320 5		370 ST+ X	
321 +		371 *	
322 XEQ IND		372 -	
X		373 ST+ 10	
323 RCL 06		374 RTN	

# Program Listings

375♦LBL E	Punish	51	
376 RCL 04			
377 X<> 13			
378 RCL 05			
379 2			
380 /			
381 ST- IND			
13			
382 RDH			
383 X<> 13			
384 "AAAI IIII		60	
... "			
385 AVIEW			
386 GTO 30			
387 .END.			
20		70	
30		80	
40		90	
50		00	

# <sup>32</sup>REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS			STATUS			
00	seed	50	SIZE 014 TOT. REG. 167			USER MODE
	Possible moves		ENG	FIX	SCI	ON OFF
	Possible moves		DEG	RAD	GRAD	
	Possible moves					
	Trial move		FLAGS			
05	Used	55	# INIT S/C	SET INDICATES		CLEAR INDICATES
	Used			05	41C first	Human first
	Used			06	41C's move	human's move
	Current move			07	mirror image	normal game
	Last move			08	first move	not first move
10	Board	60		09	Used	Used
	From					
	To					
	counter					
15		65				
20		70				
25		75				
30		80				
35		85				
40		90				
45		95				
ASSIGNMENTS						
			FUNCTION	KEY	FUNCTION	KEY

## SCATTER

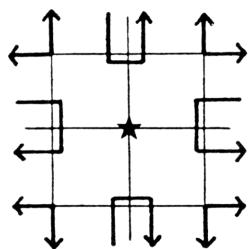
(Requires 1 Memory Module)

$N$  atoms are randomly placed in a black box with dimensions  $10 \times 10$ . No atom can be on the edge of the box. By firing particles into the box from the edges, and noting their exit locations, you attempt to find the atom positions. For a single atom, the scatters and reflections are as shown in Figure 1. Multiple atom scatters are simple extensions of this diagram: See, for examples, Figures 2 and 3. Note in particular, the back reflections of Figure 3 which arise from two combined scatters. More complex scattering and reflection are shown in Figure 4 where atom 4 causes scatter A, atom 2 causes B, 1 causes C, 4 (again) causes D, 3 causes E, and atom 1 reflects the particle back along the convoluted path. The numbering of the box grid is given in Figure 5. The 5th position on the base has coordinates 5.0, the 7th on the right is 9.7, and so on.

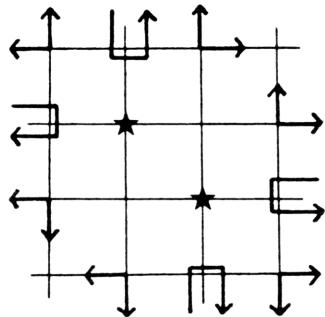
You select the value of  $N$ , and the machine places the  $N$  atoms randomly. You then fire particles from the edge: The machine tracks them and displays the output edge locations. At any time you can get the machine to confirm or reject any suspected atom location. If the guess is wrong, you are "penalized" by having the number of used particles increased by 5. The object of the game is not only to find the atoms, but to do so with the minimum number of probes.

NOTE: Although 9 atoms may be placed, a "good" game is 4 or 5.

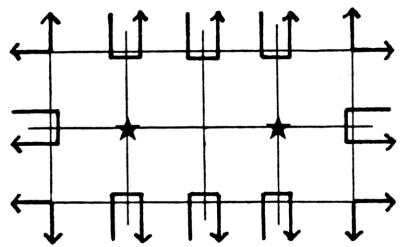
### Diagrams



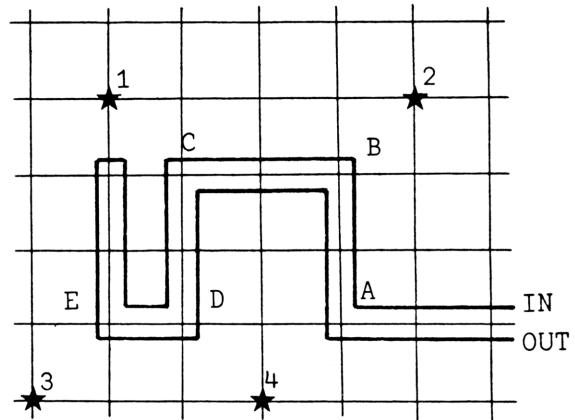
1. - SINGLE ATOM



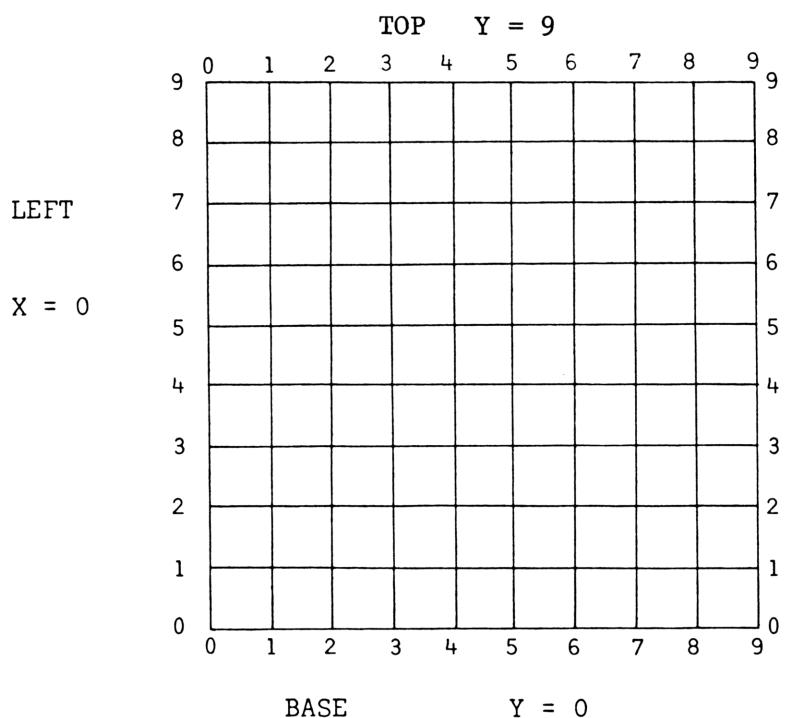
2. TWO ATOMS (1)



3. TWO ATOMS (2)



4. COMPLEX REFLECTION



5. BOX NUMBERING AND AXES

Example:

Set up and find 4 atoms.

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 022  
[XEQ] [ALPHA] SCATTER [ALPHA]  
.191062 [R/S]  
4 [R/S]  
2.0 [R/S]  
4.0 [R/S]  
3.3 [A]  
  
6.9 [R/S]  
6.8 [A]  
  
0.8 [R/S]  
  
⋮

Display:

SEED?  
NO. OF ATOMS?  
READY  
0,2  
9,2  
YES  
2 PROBES  
6,9  
NO  
8 PROBES  
9,8  
⋮

# User Instructions

# Program Listings

01♦LBL "SCA TTER"	Initialize	48 .211327 49 + 50 FRC 51 STO 00 52 8 53 * 54 1 55 + 56 INT 57 RTN	
02 SF 27 03 CF 29 04 SF 28 05 FIX 0 06 "SEED?" 07 PROMPT 08 STO 00		58♦LBL 20 59 STOP	Get an entry point
09♦LBL E 10 0 11 STO 11 12 "NO. OF ATOMS?" 13 PROMPT 14 STO 10 15 1 E3 16 / 17 1 18 + 19 STO 12	Get number of atoms	60 INT 61 STO 12 62 STO 15 63 CF 00 64 X=0? 65 SF 00 66 9 67 X=Y? 68 SF 00 69 CF 01	
20♦LBL 00 21 XEQ 10 22 XEQ 10 23 10 24 / 25 + 26 RCL 12 27 INT 28 X<>Y 29 STO 09	Place atoms	70 X=Y? 71 SF 01 72 LASTX 73 FRC 74 10 75 * 76 STO 13 77 STO 14 78 X=Y? 79 SF 01 80 1 81 ST+ 11	Select which side
30♦LBL 01 31 RCL IND Y 32 X=Y? 33 GTO 00 34 RDN 35♦LBL 09 36 DSE Y 37 GTO 01 38 STO IND 12 39 ISG 12 40 GTO 00 41 "READY" 42 AVIEW 43 GTO 20	Test to see if already filled	82♦LBL 02 83 RCL 10 84 STO 21 85 10 86 STO 17 87♦LBL 05 88 RCL IND 21 89 INT 90 LASTX 91 FRC 92 10 93 * 94 RCL 13 95 - 96 X<>Y 97 RCL 12	Get an atom
44♦LBL 10 45 RCL 00 46 9821 47 *	Get a coordinate		Decompose into X and Y
			Get X,Y distances of probe position from atom

# Program Listings

98 -		149 "F,"	
99 FS? 00		150 ARCL 14	
100 X<>Y		151 AVIEW	
101 STO 20		152 GTO 20	
102 ABS		153♦LBL 09	Perturbation
103 1	Test X distance	154 FS?C 03	
104 -		155 GTO 03	
105 X>0?		156 RCL 12	
106 GTO 09	No effect on	157 RCL 13	Deflection
107 RDN	probe	158 FS? 00	
108 STO 16		159 X<>Y	
109 FS? 01	Test Y distance	160 RCL 19	
110 CHS		161 1	
111 X<0?		162 FS? 01	
112 GTO 09	No effect	163 CHS	
113 RCL 17		164 -	
114 X<>Y	Atom hidden	165 +	
115 X>Y?	behind another?	166 FS? 00	
116 GTO 09		167 X<>Y	
117 SF 03		168 STO 13	
118 X≠Y?		169 RDN	
119 CF 03	Flag 3 on for	170 STO 12	
120 STO 17	reflection	171 FS? 00	
121 RCL 20		172 SF 02	
122 STO 18		173 SF 00	
123 RCL 16		174 FS?C 02	
124 STO 19		175 CF 00	
125 SF 02		176 CF 01	
126♦LBL 09	All atoms	177 RCL 18	
127 DSE 21	scanned?	178 X=0?	
128 GTO 05		179 GTO 03	
129 FS?C 02		180 X>0?	
130 GTO 09	Perturbation?	181 SF 01	
131 0		182 GTO 02	
132 ENTER↑		183♦LBL A	
133 9		184 RCL 10	Verify guess
134 FS? 01		185 X<>Y	
135 X<>Y		186♦LBL 04	
136 RCL 12	Get exit	187 RCL IND	
137 RCL 13	coordinates	Y	
138 FS? 00		188 "YES"	
139 X<>Y		189 X=Y?	
140 RDN		190 GTO 09	
141 FS? 00		191 RDN	
142 X<>Y		192 DSE Y	
143 STO 15		193 GTO 04	
144 RDN		194 5	
145 STO 14		195 ST+ 11	
146♦LBL 03	Display	196 "NO"	
147 CLA	coordinates	197♦LBL 09	
148 ARCL 15		198 AVIEW	

# Program Listings

199 PSE		51	
200 CLA			
201 ARCL 11			
202 "F PROBE			
S"			
203 AVIEW			
204 GTO 20			
205♦LBL C	-----		
206 RCL 10	Find and display		
207 FIX 1	all positions	60	
208 CF 28			
209♦LBL 06			
210 CLA			
211 ARCL IND			
X			
212 AVIEW			
213 PSE			
214 DSE X			
215 GTO 06			
216 SF 28			
217 FIX 0		70	
218 GTO 20			
219 .END.			
30		80	
40		90	
50		00	

# <sup>40</sup>REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS				STATUS			
00	Seed	50		SIZE .022		TOT. REG. 75	USER MODE
	Atom locations			ENG	FIX	SCI	ON OFF
05				DEG	RAD	GRAD	
		55		FLAGS			
				#	INIT S/C	SET INDICATES	CLEAR INDICATES
				00		top or bottom	left or right
				01		top or right	bottom or left
				02		used	used
				03		used	used
10	n	60					
	probe count						
	particle x						
	particle y						
	y						
15	x	65					
	Used						
	Used						
	Used						
	Used						
20	Used	70					
	Used						
25		75					
30		80					
35		85					
40		90		ASSIGNMENTS			
				FUNCTION	KEY	FUNCTION	KEY
45		95					

V

## FLIP-FLOP

Flip-Flop challenges you to change a string of 8 zeroes and 1 one (.000010000) to 1 zero and 8 ones (,111101111). Only positions containing ones can be specified for flipping. Flipping a one to a zero will automatically flip adjacent zeroes to ones and ones to zeroes. Flipping a one in either end position will flip the opposite end as well as the adjacent position.

Positions are: previous move, 123456789. Note that the position to the left of the comma always shows the last move unless the last move tried to flip a zero, at which time it will show zero.

Example:

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 013

[XEQ] [ALPHA] FLIP [ALPHA]

5

6

5

:

Display:

0,000010000

5,000101000

6,000110100

5,000001100

:

# User Instructions

SIZE:013

# Program Listings

<pre> 01♦LBL "FLI P" 02 CF 28 03 FIX 9 04 CLRG 05 9 06 STO 12 07♦LBL 00 08 10 09 RCL 12 10 CHS 11 Y↑X 12 STO IND 12 13 DSE 12 14 GTO 00 15 RCL 05 16 STO 00 17 CHS 18 STO 05 19♦LBL 01 20 RCL 00 21 .11110111 11 22 X&lt;&gt;Y 23 X=Y? 24 GTO 02 25 RCL 10 26 + 27 XEQ 05 28♦LBL 03 29 STO 12 30 STO 10 31 XEQ 07 32 ISG 11 33♦LBL 10 34 CHS 35 STO IND 12 36 ST- 00 37 9 38 RCL 12 39 X=Y? 40 GTO 09 41 1 42 X=Y? 43 GTO 04 44 ISG 12 45♦LBL 10 46 RCL IND 12 </pre>	<p>Initialize</p> <p>Update</p>	<pre> 47 CHS 48 STO IND 12 49 ST- 00 50 DSE 12 51 DSE 12 52 RCL IND 12 53 CHS 54 STO IND 12 55 ST- 00 56 GTO 01 57♦LBL 09 58 RCL 01 59 CHS 60 STO 01 61 ST- 00 62 RCL 08 63 CHS 64 STO 08 65 ST- 00 66 GTO 01 67♦LBL 04 68 RCL 02 69 CHS 70 STO 02 71 ST- 00 72 RCL 09 73 CHS 74 STO 09 75 ST- 00 76 GTO 01 77♦LBL 05 78 CF 22 79 VIEW X 80♦LBL 06 81 PSE 82 FS?C 22 83 RTN 84 GTO 06 85♦LBL 07 86 RCL IND 12 87 X&lt;0? 88 RTN 89 RCL 00 90 VIEW X 91 XEQ 06 92 GTO 03 93♦LBL 02 </pre>	<p>-----</p> <p>Right end</p> <p>-----</p> <p>Left end</p> <p>-----</p> <p>Input loop</p> <p>-----</p> <p>Valid?</p> <p>-----</p>
---	---------------------------------	--	---

# Program Listings

94 RCL 10	Game over	51	
95 +			
96 VIEW X			
97 PSE			
98 CLA			
99 FIX 0			
100 CF 29			
101 SF 28			
102 ARCL 11			
103 "FLIPS		60	
"			
104 AVIEW			
105 .END.			
20		70	
30		80	
40		90	
50		00	

# REGISTERS, STATUS, FLAGS, ASSIGNMENTS<sup>45</sup>

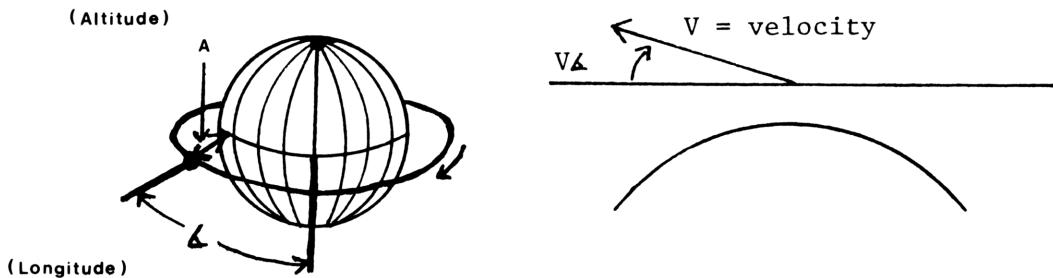
DATA REGISTERS				STATUS								
00	Current display	50		SIZE	013	TOT. REG.	48	USER MODE				
	Used			ENG	—	FIX	—	SCI	—	ON	—	OFF
	Used			DEG	—	RAD	—	GRAD	—			
	Used											
05	Used	55		FLAGS								
	Used			#	INIT S/C	SET INDICATES	CLEAR INDICATES					
	Used											
	Used											
	Used											
	Used											
10	Used	60										
# of flips												
Counter												
15		65										
20		70										
25		75										
30		80										
35		85		ASSIGNMENTS								
40		90		FUNCTION	KEY	FUNCTION	KEY					
45		95										

## ORBITAL LANDER

(Requires One Memory Module)

This program simulates a Lunar Excursion Module in orbit 100 km above the surface of the moon. The object is to execute a soft landing (velocity less than 5m/sec, at an angle not more than 5° from vertical) given a limited supply of fuel. On each move, you have the option of either free-falling for a specified period of time, or applying a specified thrust during a specified time period. Thrust is calculated and applied from your input of change in velocity over a given amount of time in a given direction from 0° to +180°. Your velocity will not actually change by this amount, of course, since gravity is also acting. You are not allowed to apply a thrust of greater than 7 Gees (69m/sec/sec of time period). If you run out of fuel, your thrust will be reduced to the fuel supply on hand. Thereafter, any thrust value you provide will be automatically changed to zero. When you pass zero altitude (i.e., land or crash), the program will calculate and display your velocity at impact. (Note to skilled pilots: try also to land at 0° longitude.)

Because the orbital equations are time-independent, the program has to convert the desired "delta-t" into a variable the equations can work with. This conversion process is not completely accurate, but the only error it introduces is that the actual duration of the jump may be slightly different from the one you specify. No positional error is introduced--you will still be on exactly the correct orbit--but you will find yourself at a slightly different point along that orbit. For example, a 2000 second jump along the initial orbit will take you almost a third of the way around the moon, but the conversion approximation will be about 10 percent shorter than an actual 2000 second jump.



Variable Conventions

**Important:** The altitude (A) is from the surface of the moon, not the center. The angle of velocity ( $V\Delta$ ) is given from horizontal, with 0° being forward and 90° straight up. Thrust angles also follow  $V\Delta$  conventions. 180° is a retrofire.

Example:

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 015

[XEQ] [ALPHA] ORBIT [ALPHA]

[R/S]

[R/S]

[R/S]

[R/S]

1000 [A]

[R/S]

[R/S]

[R/S]

[R/S]

For 10 seconds apply 7 gravities as retrofire

69 [ENTER↑] 10 [X]

180 [ENTER↑]

10 [B]

[R/S]

[R/S]

[R/S]

[R/S]

200 [A]

[R/S]

[R/S]

[R/S]

[R/S]

⋮

Display:

A=100000.00 M (altitude)

$\Delta=0.00$  (longitude)

V=1631.77 M/S (velocity)

$v\Delta=0.00$  (angle from horizontal)

F=2,000.00 (fuel)

A=99,957.06 M

$\Delta=55.65$

V=1,631.80 M/S

$v\Delta=0.00$

F=2,000.00

A=99,908.77 M

$\Delta=55.95$

V=941.88 M/S

$v\Delta=-0.59$

F=1,310.00

A=78,392.28 M

$\Delta=61.89$

V=974.77 M/S

$v\Delta=-12.14$

F=1,310.00

⋮

# User Instructions

# Program Listings

49

01♦LBL "ORB IT" 02 SF 27 03 CLRG 04 CF 05 05 2000 06 STO 00 07 1839000 08 STO 01 09 4.89663 E12 10 STO 03 11 1631.765 625 12 STO 05 13 1739000 14 STO 11 15 0 16 GTO 01 17♦LBL B 18 STO 12 19 69 20 * 21 R↑ 22 RCL 00 23 X<=Y? 24 X<>Y 25 RDN 26 X<=Y? 27 X<>Y 28 RDN 29 ST- 00 30 P-R 31 ST+ 05 32 RCL 05 33 9 34 X>Y? 35 GTO 21 36 R↑ 37 ST+ 04 38♦LBL 01 39 RCL 04 40 X↑2 41 RCL 05 42 X↑2 43 + 44 2 45 / 46 RCL 03 47 RCL 01 48 /	Initialize  Thrust  No greater than 7 gees  Compute new orbit	49 - 50 STO 06 51 RCL 01 52 RCL 05 53 * 54 STO 07 55 X↑2 56 RCL 03 57 / 58 STO 08 59 * 60 2 61 * 62 RCL 03 63 / 64 1 65 + 66 SQRT 67 STO 09 68 RCL 08 69 RCL 01 70 / 71 1 72 - 73 RCL 09 74 / 75 FIX 7 76 RND 77 ACOS 78 RCL 04 79 RCL 05 80 * 81 X>0? 82 SF 05 83 RDN 84 FS?C 05 85 CHS 86 RCL 02 87 + 88 360 89 MOD 90 STO 10 91 RCL 12 92♦LBL A 93 STO 12 94 0 95 ENTER↑ 96 ENTER↑ 97 RCL 05 98 9 99 X>Y?	Free fall
---	---	--	-----------

# Program Listings

100 GTO 21		151 RCL 02	
101 RDN		152 RCL 10	
102 RCL 12		153 -	
103 *		154 SIN	
104 RCL 03		155 *	
105 RCL 01		156 X<0?	
106 X↑2		157 SF 05	
107 /		158 RDN	
108 RCL 12		159 FS?C 05	
109 *		160 CHS	
110 2		161 ♦LBL 20	
111 /		162 STO 14	
112 RCL 04		163 RCL 13	
113 X<>Y		164 P-R	
114 -		165 STO 05	
115 RCL 12		166 RDN	
116 *		167 STO 04	
117 RCL 01		168 RCL 01	
118 +		169 RCL 11	
119 R-P		170 -	
120 RDN		171 X<0?	
121 ST+ 02		172 GTO 22	
122 RCL 08		173 ♦LBL 10	
123 RCL 09		174 FIX 2	
124 RCL 02		175 ADV	Output
125 RCL 10		176 "A="	
126 -		177 RCL 01	
127 COS		178 RCL 11	
128 *		179 -	
129 1		180 X<0?	
130 +		181 CLX	
131 /		182 ARCL X	
132 STO 01		183 "F M"	
133 RCL 03		184 AVIEW	
134 X<>Y		185 STOP	Altitude
135 /		186 "Z="	
136 RCL 06		187 RCL 02	
137 +		188 1	
138 2		189 P-R	
139 *		190 R-P	
140 SQRT		191 ARCL Y	
141 STO 13		192 AVIEW	Longitude
142 RCL 01		193 STOP	
143 *		194 "V="	
144 RCL 07		195 ARCL 13	
145 X<>Y		196 "F M/S"	
146 /		197 AVIEW	Velocity
147 FIX 7		198 STOP	
148 RND		199 "VZ="	
149 ACOS		200 ARCL 14	
150 RCL 07		201 AVIEW	Angle from horizon

# Program Listings

202 STOP		253 +
203 "F="		254 ABS
204 ARCL 00	Fuel	255 SQRT
205 AVIEW		256 CHS
206 STOP		257 STO 04
207 GTO 10		258 GTO 00
208♦LBL 21		259 .END.
209 RDN		
210 RDN	Vector sums	
211 2		60
212 /		
213 CHS		
214 RCL 05		
215 +		
216 RCL 12		
217 *		
218 X<>Y		
219 RCL 03		
220 RCL 01		
221 X↑2		70
222 /		
223 RCL 12		
224 *		
225 -		
226 ST+ 04		
227 2		
228 /		
229 CHS		
230 RCL 04		
231 +		
232 RCL 12		80
233 *		
234 RCL 01		
235 +		
236 R-P		
237 STO 01		
238 X<>Y		
239 ST+ 02		
240♦LBL 00		
241 RCL 04		
242 RCL 05		90
243 R-P		
244 STO 13		
245 X<>Y		
246 GTO 20		
247♦LBL 22		
248 ST- 01		
249 3		
250 *		
251 RCL 04		
252 X↑2	Impact	00



## PLANET LANDER

The object here is to perform a vertical descent ending in soft landing on the planet of your choosing. You select the planet before you begin by specifying the acceleration of gravity in feet per second per second. Some values are given below:

<u>Body</u>	<u><math>g(f/s^2)</math></u>
Earth	32.2
Moon	5.32
Mars	12.3
Ganymede	5.25
Pluto	7.25
Icarus (asteroid)	.394

For interest, zero and negative values of  $g$  are allowed. The fuel allocated, as calculated from  $g$ , is more than adequate for a minimum use landing. At least twice as much fuel as needed is given. Although it takes longer to calculate, 3 seconds is the time your burn is stretched over. You can only key a burn in during the zero of each three second count down.

Note that if zero or negative  $g$  is selected and you run out of fuel, you may not impact. In this case you will see "DEEP SPACE. . ." instead of the normal "VF=" for final velocity.

### Example:

Try a landing on the moon ( $g = 5.32 f/s^2$ ).

Keystrokes:

```
[XEQ] [ALPHA] SIZE [ALPHA] 007  
5.32 [XEQ] [ALPHA] GRAVITY [ALPHA]
```

Display:

```
G=5.32  
FUEL=5456  
V= -500 F/S  
A=5000 V  
THREE...  
TWO...  
ONE...  
ZERO...
```

(to free fall, just do nothing)

FUEL=5456

**Keystrokes:**

20

**Display:**

V= -516 F/S

A=3476 F

THREE...

TWO...

ONE...

ZERO...

FUEL=5436

V= -512 F/S

A=1934 F

THREE...

TWO...

ONE...

:

# User Instructions

# Program Listings

01♦LBL "GRA VITY"	Initialize	51 "THREE.. ." 52 AVIEW 53 PSE 54 "TWO..." 55 AVIEW 56 PSE 57 "ONE..." 58 AVIEW 59 PSE 60 CLX 61 "ZERO..."	Countdown
02 SF 27 03 STO 01 04 ABS 05 800 06 * 07 1200 08 + 09 STO 05		62 AVIEW 63 PSE 64 CLD 65 STO 00 66 ABS 67 RCL 03 68 X>Y? 69 RDN 70 ST- 03 71 RCL 00 72 SIGN 73 * 74 3 75 / 76 RCL 01 77 - 78 STO 04 79 RCL 06	-----
10♦LBL A 11 5000 12 STO 06 13 -500 14 STO 02 15 RCL 05 16 STO 03 17 "G=" 18 FIX 2 19 CF 29 20 ARCL 01 21 AVIEW 22 PSE 23 FIX 0	Set up initial conditions	80 * 81 2 82 * 83 RCL 02 84 X↑2 85 X<>Y 86 - 87 SF 00 88 X<0? 89 GTO 01 90 SQRT 91 RCL 02 92 + 93 CHS 94 RCL 04 95 X=0? 96 GTO 01 97 CF 00 98 / 99 3	Calculate acceleration
24♦LBL 09 25 "FUEL="	Display status		-----
26 ARCL 03 27 AVIEW 28 PSE 29 RCL 02 30 RND 31 RCL 06 32 .5 33 "V" 34 X>Y? 35 "F" 36 "F="			Calculate time to impact
37 ARCL Z 38 "F F/S" 39 AVIEW 40 PSE 41 X>Y? 42 RTN 43 "A="			
44 ARCL 06 45 "F F" 46 AVIEW 47 PSE 48 RCL 03 49 X=0?			If greater than 3 seconds use 3
50 GTO 02			

# Program Listings

100 X<>Y		150 /	t
101 X>Y?		151 X<0?	
102 RDN		152 PROMPT	
103♦LBL 01		153 RCL 01	
104 FS?C 00		154 *	
105 3	ΔV	155 ST- 02	
106 STO 00		156 0	ΔV
107 RCL 04		157 STO 06	
108 *		158 GTO 09	
109 RCL 02		159 .END.	
110 +			
111 X<> 02			
112 RCL 00			
113 X↑2			
114 RCL 04			
115 *			
116 2			
117 /			
118 X<>Y			
119 RCL 00			
120 *	ΔAltitude	70	
121 +			
122 ST+ 06			
123 GTO 09			
124♦LBL 02			
125 "DEEP SP			
ACE..."			
126 RCL 01			
127 X≠0?			
128 GTO 03			
129 0	Find velocity		
130 STO 06	calculation		
131 RCL 02			
132 X<0?			
133 GTO 09			
134 PROMPT			
135♦LBL 03			
136 RCL 01			
137 RCL 06			
138 *			
139 2			
140 *			
141 RCL 02			
142 X↑2			
143 +			
144 X<0?			
145 PROMPT			
146 SQRT			
147 RCL 02			
148 +			
149 RCL 01		00	

# <sup>58</sup>REGISTERS, STATUS, FLAGS, ASSIGNMENTS

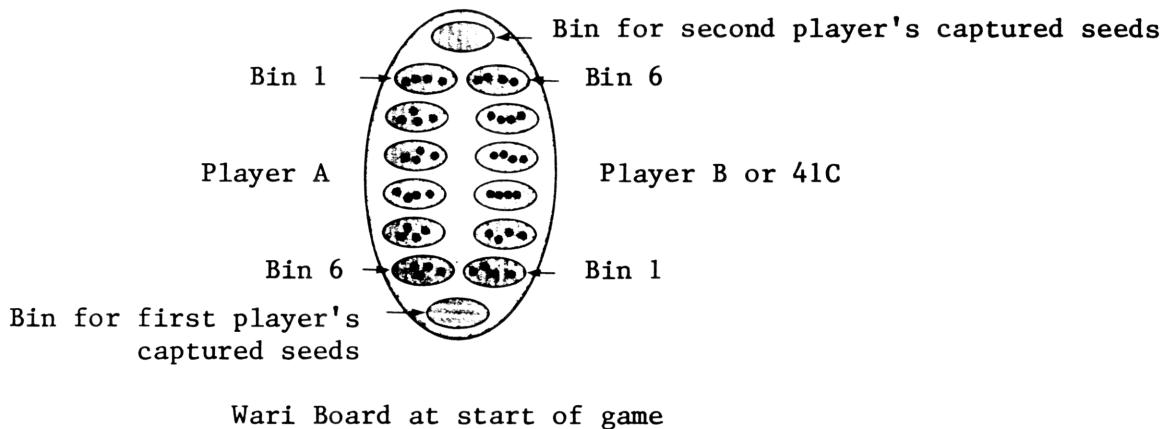
DATA REGISTERS			STATUS				
00	used	50	SIZE	007	TOT. REG.	40	USER MODE
	g		ENG		FIX		ON OFF
	Vel.		DEG		RAD		GRAD
	Fuel						
	Used						
05	Initial fuel	55	FLAGS				
	Alt		#	INIT S/C	SET INDICATES	CLEAR INDICATES	
10		60					
15		65					
20		70					
25		75					
30		80					
35		85					
ASSIGNMENTS							
			FUNCTION	KEY	FUNCTION	KEY	
40		90					
45		95					

## WARI

(This program requires one Memory Module)

Wari\* is a board game which has been played for at least several centuries in various forms throughout Africa. The game is played on a board containing (generally) twelve small pits or bins, and two large pits. Forty-eight beads, seeds, or other counters are moved and captured according to certain rules.

The Wari board shown here is set up to begin a game.



Each player in turn removes all the counters from one bin on his side and distributes them one-at-a-time into successive bins moving counterclockwise, skipping the two bins which are for storing captured counters. If the last counter drops into an opponent's hole containing one or two counters, the contents of that hole are captured and placed in the player's scoring pit. Counters in an unbroken sequence of two- and three-counter bins on the opponent's side clockwise from the captured bin are also captured. If a bin contains twelve counters or more, that bin is skipped when the counters from that bin are distributed.

The above rules are implemented in the calculator program. Special rules, such as prohibiting moves which remove all of the opponent's counters, were deemed to be variations of the basic game and were not programmed. It is possible to come to a situation where a few counters will circulate forever. In this case each player claims the counters on his side.

To make a play on the calculator Wari board, the player specifies the bin he wants to move by keying in a number from 1 to 6 and then pushing either [A] or [B] (for player A or B). The machine then moves the counters from the specified bin according to the rules. To play against the calculator, signal to the calculator to move by pressing [C]. The calculator will then move player B's counters.

\*Also known as Man-Kalah, Awari, and many other names.

When one of the sides of the board is displayed, as designated by leading A or B, it is as if you moved around to that side of the board. In other words, bin 1 for either players side is always to the left and counterclockwise always to the right. If you are looking at side A, [R/S] will get you side B. If you are looking at side B, [R/S] will get you the score. If you are looking at the score, [R/S] will get you side A.

**Example:**

Start a game and have the calculator make the first move.

**Keystrokes:**

[XEQ] [ALPHA] SIZE [ALPHA] 018

[XEQ] [ALPHA] WARI [ALPHA]

.9977663333 [R/S]

[R/S]

[R/S]

[C]

[R/S]

6 [A]

[R/S]

[C]

[R/S]

2 [A]

[R/S]

[C]

[R/S]

[R/S]

:

**Display:**

SEED?

A 4,4,4,4,4,4

B 4,4,4,4,4,4

A=0, B=0

A 5,5,4,4,4,4

B 4,4,4,0,5,5

A 5,5,4,4,4,0

B 5,5,5,1,5,5

A 6,6,4,4,4,0

B 5,5,0,2,6,6

A 6,0,5,5,5,1

B 6,6,0,2,6,6,

A 7,1,6,6,6,0

B 6,6,0,2,6,0

A=0, B=2

:

# User Instructions

# Program Listings

<pre> 01♦LBL "WAR I" 02 SF 27 03 CF 29 04 FIX 0 05 "SEED?" 06 PROMPT 07 STO 00 08♦LBL E 09 1.012 10 CLRG 11 4 12♦LBL 06 13 STO IND Y 14 ISG Y 15 GTO 06 16 GTO 50 17♦LBL C 18 CF 05 19 0 20 STO 17 21 STO 15 22 2 23 XEQ 51 24 CF 06 25 X=0? 26 SF 06 27 CF 07 28 7.012 29 STO 16 30♦LBL 05 31 RCL 16 32 INT 33 RCL IND 16 34 X=0? 35 GTO 00 36 RCL X 37 12 38 / 39 INT 40 STO T 41 + 42 + 43 12 44 MOD 45 X=0? 46 X&lt;&gt; L 47 7 48 X&lt;=Y? </pre>	<p>Initialize</p> <p>41C move</p> <p>Offense</p> <p>41C pit #</p> <p>41C counters</p> <p>Player pit #</p>	<pre> 49 GTO 00 50 X&lt;&gt;Y 51 R↑ 52 CHS 53 RCL IND Y 54 X&lt;=Y? 55 GTO 00 56 X&lt;&gt;Y 57 3 58 + 59 X&lt;=Y? 60 GTO 00 61 SF 07 62 RCL Z 63 RCL 17 64 X&gt;Y? 65 GTO 00 66 X≠Y? 67 GTO 01 68 FS? 06 69 GTO 00 70♦LBL 01 71 RDN 72 STO 17 73 RCL 16 74 STO 15 75♦LBL 00 76 ISG 16 77 GTO 05 78 FC?C 07 79 GTO 00 80 RCL 15 81 INT 82 GTO A 83♦LBL 00 84 6 85 STO 16 86 0 87 STO 15 88♦LBL 04 89 RCL 16 90 RCL IND 16 91 X=0? 92 GTO 00 93 RCL X 94 12 95 / 96 INT 97 X&gt;0? </pre>	<p>Player counters</p> <p>Found an offensive move</p>	<p>Defense</p> <p>Player's pit #</p> <p>Player's counters</p>
--	---	---	---	---

# Program Listings

98 GTO 00		147 RCL 16	
99 RDN		148 X=Y?	
100 +		149 GTO 01	
101 12		150 RDN	
102 MOD		151 STO 15	
103 X=0?		152 GTO 03	
104 LASTX	41C pit #	153♦LBL 01	
105 7		154 SF 05	
106 X>Y?		155 "NO MOVE"	
107 GTO 00		156 AVIEW	
108 RCL IND	41C counters	157 STOP	
Y		158 GTO 50	
109 X=0?		159♦LBL 00	
110 GTO 00		160 RCL 15	
111 3		161♦LBL B	-----
112 X<=Y?		162 6	Player B
113 GTO 00		163 +	
114 SF 07		164 CF 05	
115 RT		165♦LBL A	
116 RCL 15		166 STO 15	
117 X<=Y?		167 STO 17	Player A
118 X<>Y		168 RCL IND	
119 STO 15		15	
120♦LBL 00		169 X=0?	
121 DSE 16		170 GTO 50	
122 GTO 04		171 STO 16	
123 FC?C 07		172 ST- IND	
124 GTO 00		15	
125 RCL 15	Defensive move	173♦LBL 08	
126 GTO A	found	174 RCL 17	Distribute
127♦LBL 00		175 1	counters
128 6		176 +	
129 XEQ 51		177 12	
130 1		178 MOD	
131 +		179 X=0?	
132 STO 15		180 LASTX	
133 STO 16		181 STO 17	
134♦LBL 03	Random move	182 RCL 15	
135 6		183 X=Y?	
136 +		184 GTO 08	
137 RCL IND		185 1	
X		186 ST+ IND	
138 X≠0?		17	
139 GTO 00		187 DSE 16	
140 RCL 15		188 GTO 08	
141 1		189♦LBL 07	
142 -		190 RCL 17	
143 6		191 7	
144 MOD		192 FS? 05	
145 X=0?		193 GTO 01	
146 LASTX			

# Program Listings

194 X<=Y?		242♦LBL 00	
195 GTO 50		243 ARCL IND	
196 GTO 00		X	
197♦LBL 01		244 ISG X	
198 X>Y?		245 GTO 09	
199 GTO 50		246 RTN	
200♦LBL 00	Capture pits	247♦LBL 51	
201 RCL IND		248 RCL 00	
17		249 9821	
202 2		250 *	
203 X>Y?		251 .211327	
204 GTO 50		252 +	
205 RDN		253 FRC	
206 4		254 STO 00	
207 X<=Y?		255 *	
208 GTO 50		256 INT	
209 RDN		257 RTN	
210 FS? 05		258 .END.	
211 ST+ 13		70	
212 FC? 05			
213 ST+ 14			
214 ST- IND			
17			
215 RCL 17			
216 1			
217 -			
218 12			
219 MOD			
220 X=0?			
221 LASTX			
222 STO 17		80	
223 GTO 07			
224♦LBL 50			
225 SF 05	Output		
226 "A "			
227 1.006			
228 XEQ 00			
229 PROMPT	A board		
230 "B "			
231 7.012			
232 XEQ 00	B board	90	
233 PROMPT			
234 "A="			
235 ARCL 13			
236 "F, B=			
"			
237 ARCL 14	Score		
238 PROMPT			
239 GTO 50			
240♦LBL 09			
241 "F,"		00	

-----  
Random number  
generator

# REGISTERS, STATUS, FLAGS, ASSIGNMENTS<sup>65</sup>

DATA REGISTERS			STATUS				
			SIZE	018	TOT. REG.	82	USER MODE
			ENG	FIX	SCI	ON	OFF
			DEG	RAD	GRAD		
FLAGS							
			#	INIT S/C	SET INDICATES	CLEAR INDICATES	
00	Seed	50		05	A's move	B's move	
	Bin A1			06	Used		
	A2			07	Found one to attack		
05	A4						
	A5	55					
	A6						
	Bin B1						
	B2						
	B3						
10	B4	60					
	B5						
	B6						
	Score A						
	Score B						
15	Used	65					
	Used						
	Used						
20		70					
25		75					
30		80					
35		85					
ASSIGNMENTS							
			FUNCTION	KEY	FUNCTION	KEY	
40		90					
45		95					

## SIMON

This game exercises your memory by presenting longer and longer sequences of random numbers. You try to remember and key in each sequence. Flag settings may be varied to change the difficulty.

Example: Use a seed of  $\pi$  for the random number generation to duplicate this game.

**Keystrokes:**

```
[XEQ] [ALPHA] SIZE [ALPHA] 004
[///] [ $\pi$ ] [STO] 00
[XEQ] [ALPHA] SIMON [ALPHA]
3 [R/S]
```

1 [R/S]

9 [R/S]

346 [R/S]

**Display:**

```
HOW MANY?
1 (sequence)
```

NUMBERS?

YES: 1

```
9 (sequence)
```

2

NUMBERS?

NO: 92, NOT 9

3

```
4 (sequence)
```

6

NUMBERS?

YES: 346

YOU MISSED 1

# User Instructions

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY	SIZE:004
1	Key in program and store seed ( $0 \leq s < 1$ )	s	[STO] 00		
2	Begin a new game.		[XEQ] SIMON	HOW MANY?	
3	Key in the longest sequence you desire	n* ( $\leq 10$ )	[R/S]	(SEQUENCE) NUMBERS	
4	Repeat the sequence you just saw	sequence	[R/S]	YES (or) NO	
5	Step 4 is repeated until you win or lose			YOU WIN	
				YOU MISSED ( )	
	Note 1: You can set flag 0 (SF00) to use longer and longer pieces of the same sequence. This version of the game is easier for young children.				
	Note 2: You can clear flag 26 (CF26) to suppress the tone and make the sequences pass more quickly.  Some people find them easier to memorize this way.				
	*You can start with a sequence longer than 1 digit by keying in a number of the form 100 a+b where a is one less than the length of the first sequence you want and b is the maximum length. For example, 2006 would yield sequences of lengths 3, 4, 5 and 6.				

# Program Listings

01♦LBL "R"		49 BEEP	
02 FS? 00	Random number generator	50 "YOU WIN	Win message
03 RTN		"	
04 RCL 00		51 AVIEW	
05 9821		52 RTN	
06 *		53♦LBL "TON	
07 .211327		ES"	
08 +		54 RCL 01	
09 FRC		55 INT	
10 1.1111		56 STO 03	Set up tone counter
11 *		57 RCL 00	
12 FRC		58 FRC	
13 STO 00		59♦LBL 03	Begin tone loop
14 RTN		60 10	
15♦LBL "SIM		61 *	
ON"	Set display format	62 INT	
16 FIX 0		63 VIEW X	
17 CF 29		64 TONE IND	
18 CF 06	Clear error flag	X	
19 FS?C 00		65 LASTX	
20 SF 07		66 FRC	Decrement tone count
21 XEQ "R"		67 DSE 03	
22 FS?C 07		68 GTO 03	
23 SF 00	Save status of FOO	69 RTN	Repeat tone loop
24 "HOW MAN		70♦LBL "NO"	
Y?"		71 SF 06	
25 PROMPT	Ask for maximum length	72 TONE 2	"Un-oh" sound
26 1 E3		73 TONE 0	
27 /		74 "NO: "	
28 1		75 ARCL X	
29 +		76 "F, NOT	
30 STO 01	Set up counters	"	
31 CLX		77 ARCL Y	Increment error count
32 STO 02		78 AVIEW	
33♦LBL 10		79 PSE	
34 RCL 01	Get a sequence	80 1	Increment counter
35 INT		81 ST+ 02	
36 XEQ "R"	Display the sequence	82 ISG 01	Repeat loop
37 XEQ "TON		83 GTO 10	
ES"		84♦LBL 01	
38 XEQ "?"	Ask player what he saw	85 "YOU MIS	Error message
39 FS? 05		SED "	
40 GTO "NO"	If wrong, branch to "NO"	86 ARCL 02	
41 "YES: "		87 CF 06	
42 ARCL X		88 AVIEW	
43 AVIEW	Increment counter	89 RTN	
44 ISG 01		90♦LBL "?"	
45 GTO 10	Repeat loop	91 "NUMBERS	Ask player to input numbers seen
46 FS?C 06		?"	
47 GTO 01	If any errors, skip win message	92 PROMPT	
48 BEEP		93 CF 05	

# Program Listings

94 RCL 00		51	
95 RCL 01			
96 INT			
97 10↑X			
98 *			
99 INT	If correct, then done		
100 X=Y?			
101 RTN			
102 SF 05	Otherwise, set error flag	60	
103 RTN			
104 .END.			
20		70	
30		80	
40		90	
50		00	

# <sup>70</sup> REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS				STATUS			
00	Random sequence	50		SIZE <u>004</u> TOT. REG. <u>40</u> USER MODE			
	length counter			ENG <u>0</u> SCI <u>OFF</u> ON <u>OFF</u>			
	error counter			DEG <u>RAD</u> GRAD			
	tones counter						
05		55		FLAGS			
				# INIT S/C	SET INDICATES	CLEAR INDICATES	
				00	same sequence	different sequence	
				06	error has occurred	no error yet	
				07	temporarily matches	flag 00	
10		60		26	tones	no tones	
15		65					
20		70					
25		75					
30		80					
35		85					
				ASSIGNMENTS			
40		90		FUNCTION	KEY	FUNCTION	KEY
45		95					

**HEWLETT-PACKARD**

**HP-41C**

**USERS' LIBRARY SOLUTIONS**

**Bar Codes**

**Games**

## **GAMES**

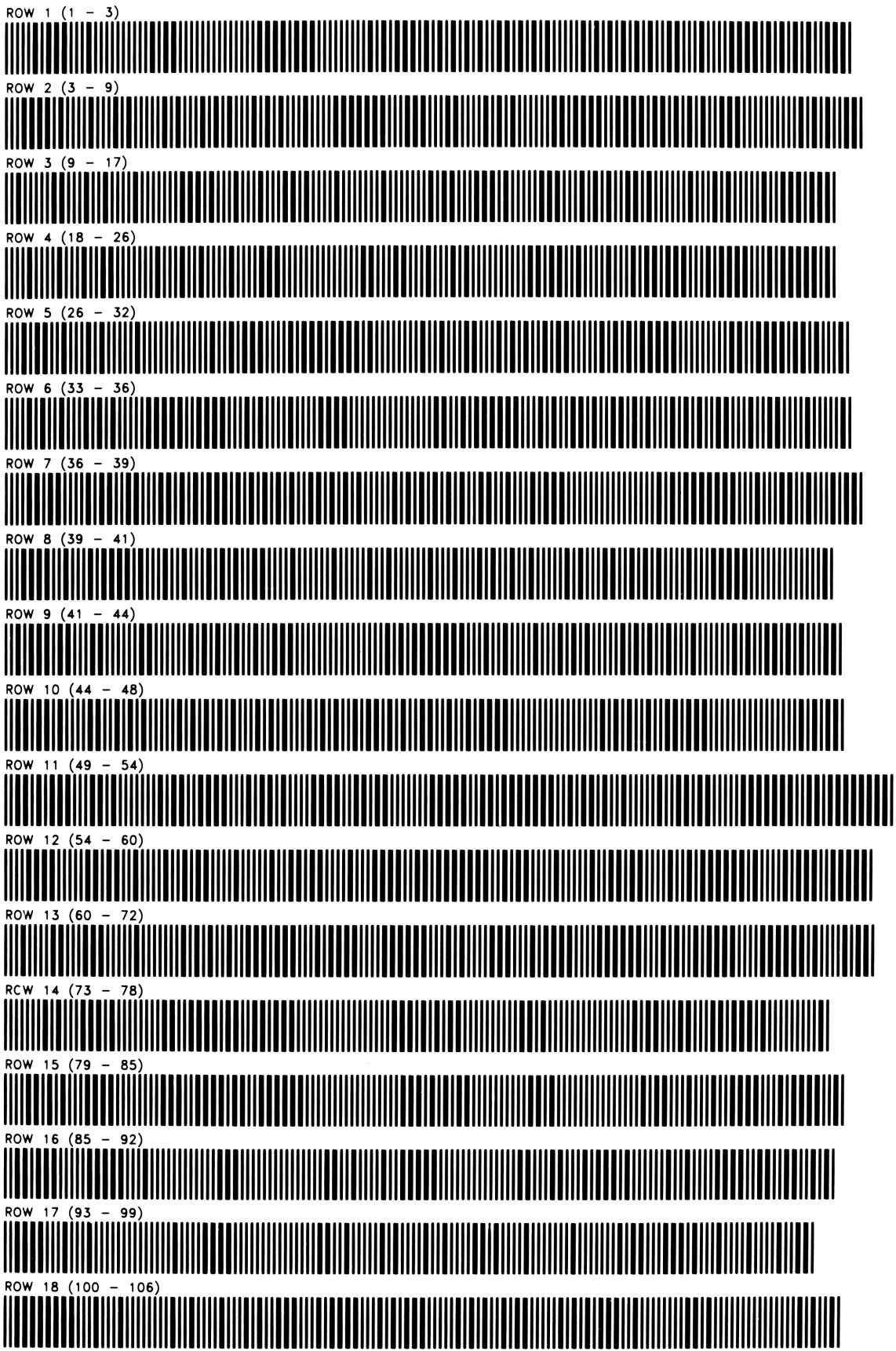
HUNT THE WUMPUS .....	1
3-D TIC TAC TOE .....	3
ROBOT TRAP .....	7
HEXAPAWN .....	10
SCATTER .....	15
FLIP-FLOP .....	17
ORBITAL LANDER .....	18
PLANET LANDER .....	20
WARI .....	22
SIMON .....	24

## **NOTICE**

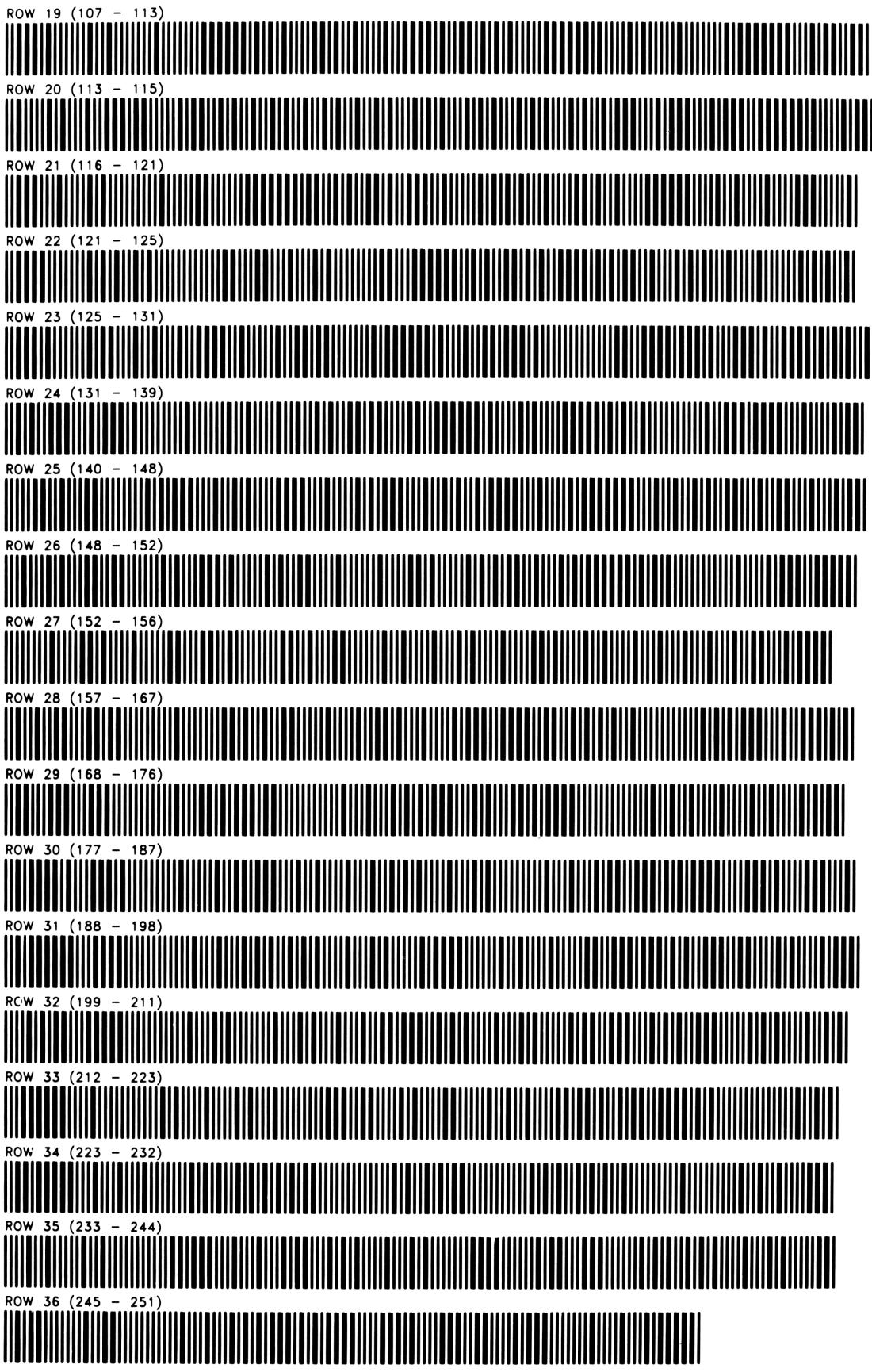
The program material contained herein is supplied without representation or warranty of any kind. Hewlett-Packard Company therefore assumes no responsibility and shall have no liability, consequential or otherwise, of any kind arising from the use of this program material or any part thereof.

HUNT THE WUMPUS

PROGRAM REGISTERS NEEDED: 67



## HUNT THE WUMPUS



**3-D TIC TAC TOE**

**PROGRAM REGISTERS NEEDED: 108**

ROW 1 (1 – 4)



ROW 2 (4 – 9)



ROW 3 (9 – 16)



ROW 4 (17 – 24)



ROW 5 (24 – 31)



ROW 6 (31 – 34)



ROW 7 (35 – 42)



ROW 8 (43 – 49)



ROW 9 (49 – 56)



ROW 10 (57 – 62)



ROW 11 (63 – 68)



ROW 12 (69 – 77)



ROW 13 (78 – 86)



RCW 14 (87 – 97)



ROW 15 (98 – 107)



ROW 16 (108 – 117)



ROW 17 (118 – 124)



ROW 18 (125 – 133)



## 3-D TIC TAC TOE

ROW 19 (133 – 140)



ROW 20 (141 – 147)



ROW 21 (147 – 154)



ROW 22 (154 – 162)



ROW 23 (162 – 170)



ROW 24 (170 – 178)



ROW 25 (178 – 186)



ROW 26 (187 – 197)



ROW 27 (198 – 203)



ROW 28 (203 – 214)



ROW 29 (215 – 221)



ROW 30 (222 – 229)



ROW 31 (229 – 238)



RCW 32 (238 – 246)



ROW 33 (246 – 252)



ROW 34 (253 – 259)



ROW 35 (260 – 265)

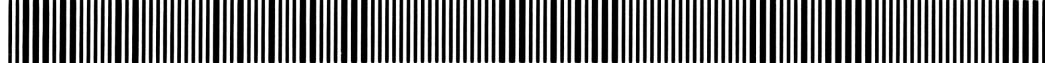


ROW 36 (266 – 273)

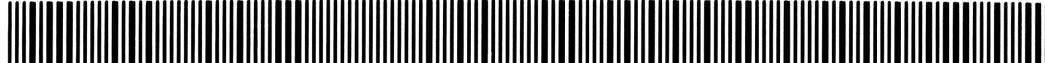


## 3-D TIC TAC TOE

ROW 37 (274 – 279)



ROW 38 (280 – 287)



ROW 39 (287 – 294)



ROW 40 (294 – 300)



ROW 41 (301 – 309)



ROW 42 (310 – 318)



ROW 43 (319 – 326)



ROW 44 (327 – 334)



ROW 45 (335 – 342)



ROW 46 (342 – 350)



ROW 47 (351 – 359)



ROW 48 (359 – 363)



ROW 49 (363 – 368)



ROW 50 (368 – 373)



ROW 51 (373 – 382)



ROW 52 (382 – 392)



ROW 53 (393 – 404)



ROW 54 (404 – 412)

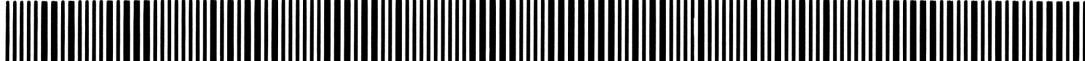


## 3-D TIC TAC TOE

ROW 55 (413 - 420)



ROW 56 (421 - 423)



ROW 57 (424 - 428)



ROW 58 (428 - 432)



ROBOT TRAP

PROGRAM REGISTERS NEEDED: 71

ROW 1 (1 – 3)



ROW 2 (3 – 7)



ROW 3 (7 – 14)



ROW 4 (15 – 17)



ROW 5 (18 – 25)



ROW 6 (26 – 31)



ROW 7 (32 – 41)



ROW 8 (42 – 46)



ROW 9 (46 – 53)



ROW 10 (54 – 62)



ROW 11 (62 – 69)



ROW 12 (70 – 73)



ROW 13 (73 – 77)



ROW 14 (78 – 80)



ROW 15 (80 – 86)



ROW 16 (87 – 87)



ROW 17 (88 – 95)



ROW 18 (96 – 103)



## ROBOT TRAP

ROW 19 (104 – 110)



ROW 20 (110 – 116)



ROW 21 (117 – 122)



ROW 22 (123 – 130)



ROW 23 (131 – 138)



ROW 24 (139 – 146)



ROW 25 (147 – 154)



ROW 26 (155 – 164)



ROW 27 (165 – 174)



ROW 28 (175 – 182)



ROW 29 (183 – 187)



ROW 30 (187 – 191)



ROW 31 (191 – 198)



ROW 32 (198 – 202)



ROW 33 (202 – 207)



ROW 34 (207 – 212)



ROW 35 (213 – 222)



ROW 36 (222 – 231)



ROBOT TRAP

ROW 37 (232 - 238)



ROW 38 (238 - 245)



ROW 39 (245 - 245)



HEXAPAWN

PROGRAM REGISTERS NEEDED: 154

ROW 1 (1 - 2)



ROW 2 (2 - 5)



ROW 3 (5 - 10)



ROW 4 (10 - 12)



ROW 5 (12 - 15)



ROW 6 (15 - 18)



ROW 7 (19 - 24)



ROW 8 (25 - 30)



ROW 9 (30 - 35)



ROW 10 (35 - 42)



ROW 11 (42 - 48)



ROW 12 (48 - 55)



ROW 13 (56 - 62)



ROW 14 (63 - 72)



ROW 15 (72 - 80)



ROW 16 (81 - 87)



ROW 17 (87 - 89)



ROW 18 (89 - 91)



# HEXAPAWN

ROW 19 (91 – 93)



ROW 20 (93 – 94)



ROW 21 (95 – 96)



ROW 22 (96 – 98)



ROW 23 (98 – 100)



ROW 24 (100 – 102)



ROW 25 (102 – 104)



ROW 26 (104 – 106)



ROW 27 (107 – 109)



ROW 28 (109 – 111)



ROW 29 (112 – 114)



ROW 30 (114 – 116)



ROW 31 (117 – 119)



RCW 32 (119 – 122)



ROW 33 (122 – 124)



ROW 34 (124 – 126)



ROW 35 (126 – 128)



ROW 36 (128 – 130)



# HEXAPAWN

ROW 37 (130 – 133)



ROW 38 (133 – 135)



ROW 39 (136 – 138)



ROW 40 (138 – 140)



ROW 41 (140 – 142)



ROW 42 (142 – 144)



ROW 43 (144 – 146)



ROW 44 (146 – 148)



ROW 45 (149 – 154)



ROW 46 (154 – 156)



ROW 47 (156 – 158)



ROW 48 (158 – 160)



ROW 49 (160 – 162)



ROW 50 (162 – 164)



ROW 51 (164 – 166)



ROW 52 (167 – 169)



ROW 53 (169 – 171)



ROW 54 (171 – 173)



# HEXAPAWN

ROW 55 (173 – 175)



ROW 56 (175 – 177)



ROW 57 (178 – 180)



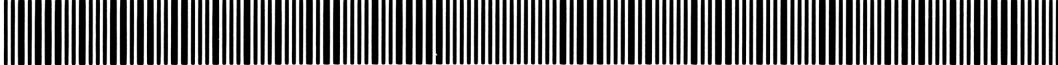
ROW 58 (180 – 182)



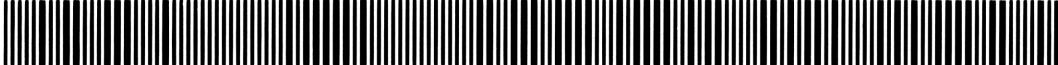
ROW 59 (182 – 184)



ROW 60 (184 – 187)



ROW 61 (187 – 194)



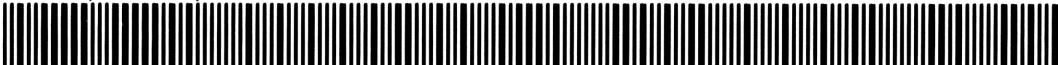
ROW 62 (195 – 206)



ROW 63 (207 – 215)



ROW 64 (215 – 223)



ROW 65 (224 – 234)



ROW 66 (235 – 243)



ROW 67 (244 – 252)



ROW 68 (252 – 256)



ROW 69 (257 – 262)



ROW 70 (263 – 273)



ROW 71 (274 – 283)



ROW 72 (283 – 293)



# HEXAPAWN

ROW 73 (294 – 301)



ROW 74 (302 – 307)



ROW 75 (308 – 316)



ROW 76 (317 – 328)



ROW 77 (329 – 336)



ROW 78 (337 – 348)



ROW 79 (349 – 359)



ROW 80 (360 – 370)



ROW 81 (371 – 380)



ROW 82 (381 – 384)



ROW 83 (384 – 387)



SCATTER

PROGRAM REGISTERS NEEDED: 54

ROW 1 (1 - 2)



ROW 2 (3 - 7)



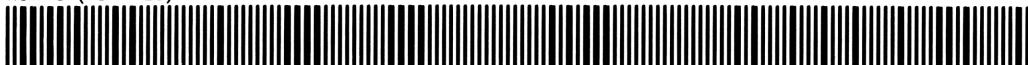
ROW 3 (8 - 12)



ROW 4 (12 - 17)



ROW 5 (18 - 25)



ROW 6 (26 - 36)



ROW 7 (36 - 41)



ROW 8 (41 - 47)



ROW 9 (48 - 54)



ROW 10 (55 - 65)



ROW 11 (65 - 74)



ROW 12 (74 - 84)



ROW 13 (84 - 92)



ROW 14 (93 - 103)



ROW 15 (104 - 112)



ROW 16 (113 - 120)



ROW 17 (121 - 127)



ROW 18 (128 - 136)



# SCATTER

ROW 19 (137 – 147)



ROW 20 (148 – 154)



ROW 21 (154 – 162)



ROW 22 (163 – 172)



ROW 23 (173 – 179)



ROW 24 (180 – 188)



ROW 25 (188 – 195)



ROW 26 (195 – 202)



ROW 27 (202 – 206)



ROW 28 (207 – 215)



ROW 29 (215 – 219)



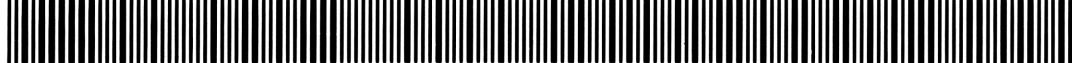
FLIP-FLOP

PROGRAM REGISTERS NEEDED: 26

ROW 1 (1 - 4)



ROW 2 (5 - 14)



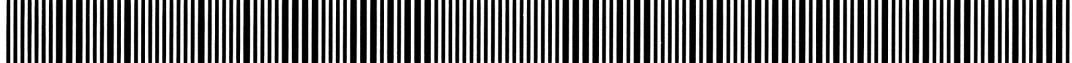
ROW 3 (14 - 21)



ROW 4 (21 - 27)



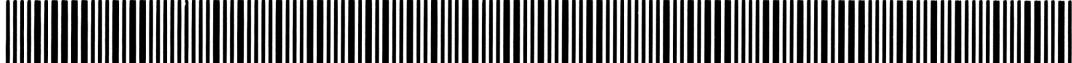
ROW 5 (28 - 36)



ROW 6 (36 - 45)



ROW 7 (46 - 52)



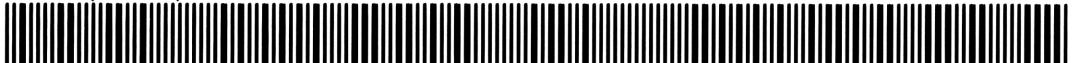
ROW 8 (53 - 61)



ROW 9 (62 - 71)



ROW 10 (72 - 80)



ROW 11 (81 - 90)



ROW 12 (90 - 98)



ROW 13 (99 - 103)



ROW 14 (103 - 105)



ORBITAL LANDER

PROGRAM REGISTERS NEEDED: 52

ROW 1 (1 - 4)



ROW 2 (4 - 7)



ROW 3 (8 - 11)



ROW 4 (11 - 13)



ROW 5 (13 - 19)



ROW 6 (19 - 30)



ROW 7 (31 - 39)



ROW 8 (40 - 52)



ROW 9 (53 - 65)



ROW 10 (66 - 77)



ROW 11 (78 - 88)



ROW 12 (88 - 98)



ROW 13 (99 - 109)



ROW 14 (110 - 121)



ROW 15 (122 - 134)



ROW 16 (135 - 147)



ROW 17 (147 - 158)



ROW 18 (159 - 169)



## ORBITAL LANDER

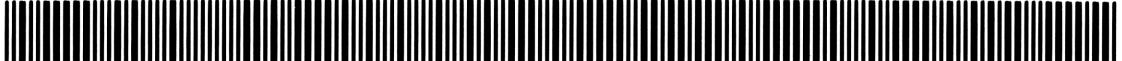
ROW 19 (170 – 177)



ROW 20 (178 – 186)



ROW 21 (186 – 194)



ROW 22 (195 – 199)



ROW 23 (199 – 207)



ROW 24 (207 – 218)



ROW 25 (219 – 230)



ROW 26 (231 – 242)



ROW 27 (243 – 251)



ROW 28 (252 – 259)



PLANET LANDER

PROGRAM REGISTERS NEEDED: 39

ROW 1 (1 - 2)



ROW 2 (3 - 10)



ROW 3 (10 - 16)



ROW 4 (17 - 23)



ROW 5 (24 - 30)



ROW 6 (31 - 37)



ROW 7 (37 - 43)



ROW 8 (43 - 50)



ROW 9 (51 - 54)



ROW 10 (54 - 57)



ROW 11 (57 - 62)



ROW 12 (63 - 74)



ROW 13 (75 - 87)



ROW 14 (87 - 97)



ROW 15 (97 - 108)



ROW 16 (109 - 120)



ROW 17 (121 - 125)

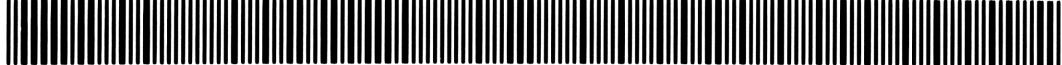


ROW 18 (125 - 130)



PLANET LANDER

ROW 19 (131 – 142)



ROW 20 (143 – 155)



ROW 21 (155 – 159)



WARI

PROGRAM REGISTERS NEEDED: 64

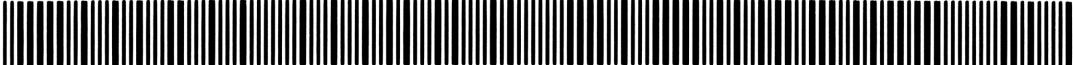
ROW 1 (1 - 4)



ROW 2 (4 - 9)



ROW 3 (9 - 16)



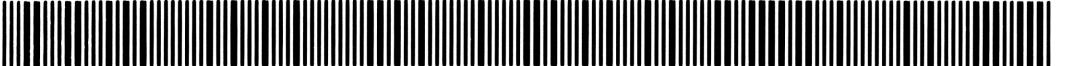
ROW 4 (16 - 23)



ROW 5 (23 - 28)



ROW 6 (29 - 36)



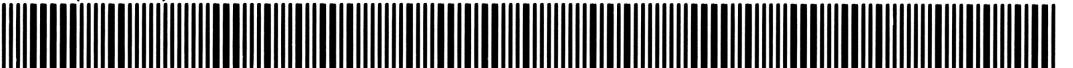
ROW 7 (37 - 46)



ROW 8 (46 - 55)



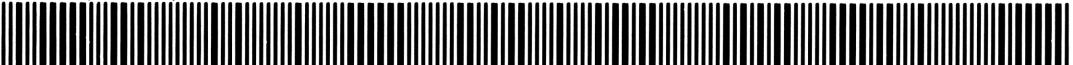
ROW 9 (56 - 64)



ROW 10 (65 - 72)



ROW 11 (73 - 80)



ROW 12 (81 - 89)



ROW 13 (90 - 98)



ROW 14 (98 - 108)



ROW 15 (108 - 117)



ROW 16 (118 - 126)



ROW 17 (126 - 134)



ROW 18 (135 - 145)



WARI

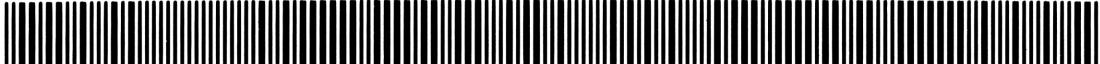
ROW 19 (146 – 154)



ROW 20 (155 – 158)



ROW 21 (159 – 167)



ROW 22 (168 – 174)



ROW 23 (175 – 184)



ROW 24 (185 – 192)



ROW 25 (193 – 199)



ROW 26 (200 – 208)



ROW 27 (208 – 215)



ROW 28 (215 – 224)



ROW 29 (224 – 228)



ROW 30 (228 – 232)



ROW 31 (233 – 236)



ROW 32 (236 – 243)



ROW 33 (243 – 249)



ROW 34 (250 – 256)



ROW 35 (257 – 258)



SIMON

PROGRAM REGISTERS NEEDED: 36

ROW 1 (1 - 5)



ROW 2 (6 - 10)



ROW 3 (10 - 15)



ROW 4 (15 - 20)



ROW 5 (21 - 24)



ROW 6 (24 - 31)



ROW 7 (32 - 37)



ROW 8 (37 - 41)



ROW 9 (41 - 47)



ROW 10 (47 - 52)



ROW 11 (53 - 57)



ROW 12 (58 - 67)



ROW 13 (67 - 72)



ROW 14 (72 - 76)



ROW 15 (76 - 82)



ROW 16 (82 - 85)



ROW 17 (85 - 90)



ROW 18 (90 - 93)

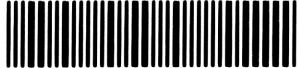


SIMON

ROW 19 (94 - 104)



ROW 20 (104 - 104)





## Hewlett-Packard Software

In terms of power and flexibility, the problem-solving potential of the HP-41C programmable calculator is nearly limitless. And in order to see the practical side of this potential, HP has different types of software to help save you time and programming effort. Every one of our software solutions has been carefully selected to effectively increase your problem-solving potential. Chances are, we already have the solutions you're looking for.

## Application Pacs

To increase the versatility of your HP-41C, HP has an extensive library of "Application Pacs". These programs transform your HP-41C into a specialized calculator in seconds. Included in these pacs are detailed manuals with examples, miniature plug-in Application Modules, and keyboard overlays. Every Application Pac has been designed to extend the capabilities of the HP-41C.

You can choose from:

<b>Aviation</b>	<b>Structural Analysis</b>	<b>Home Management</b>
<b>Clinical Lab</b>	<b>Surveying</b>	<b>Machine Design</b>
<b>Circuit Analysis</b>	<b>Securities</b>	<b>Navigation</b>
<b>Financial Decisions</b>	<b>Statistics</b>	<b>Real Estate</b>
<b>Mathematics</b>	<b>Stress Analysis</b>	<b>Thermal and Transport Science</b>
	<b>Games</b>	

## Users' Library

The Users' Library provides the best programs from contributors and makes them available to you. By subscribing to the HP-41C Users' Library you'll have at your fingertips literally hundreds of different programs from many different application areas.

## \* Users' Library Solutions Books

Hewlett-Packard offers a wide selection of Solutions Books complete with user instructions, examples, and listings. These solution books will complement our other software offerings and provide you with a valuable tool for program solutions.

You can choose from:

<b>Business Stat/Marketing/Sales</b>	<b>Civil Engineering</b>
<b>Home Construction Estimating</b>	<b>Heating, Ventilating &amp; Air Conditioning</b>
<b>Lending, Saving and Leasing</b>	<b>Mechanical Engineering</b>
<b>Real Estate</b>	<b>Solar Engineering</b>
<b>Small Business</b>	<b>Calendars</b>
<b>Geometry</b>	<b>Cardiac/Pulmonary</b>
<b>High-Level Math</b>	<b>Chemistry</b>
<b>Test Statistics</b>	<b>Games</b>
<b>Antennas</b>	<b>Optometry I (General)</b>
<b>Chemical Engineering</b>	<b>Optometry II (Contact Lens)</b>
<b>Control Systems</b>	<b>Physics</b>
<b>Electrical Engineering</b>	<b>Surveying</b>
<b>Fluid Dynamics and Hydraulics</b>	

\* Some books require additional memory modules to accomodate all programs.

## **GAMES**

HUNT THE WUMPUS  
3-D TIC TAC TOE  
ROBOT TRAP  
HEXAPAWN  
SCATTER  
FLIP-FLOP  
ORBITAL LANDER  
PLANET LANDER  
WARI  
SIMON

