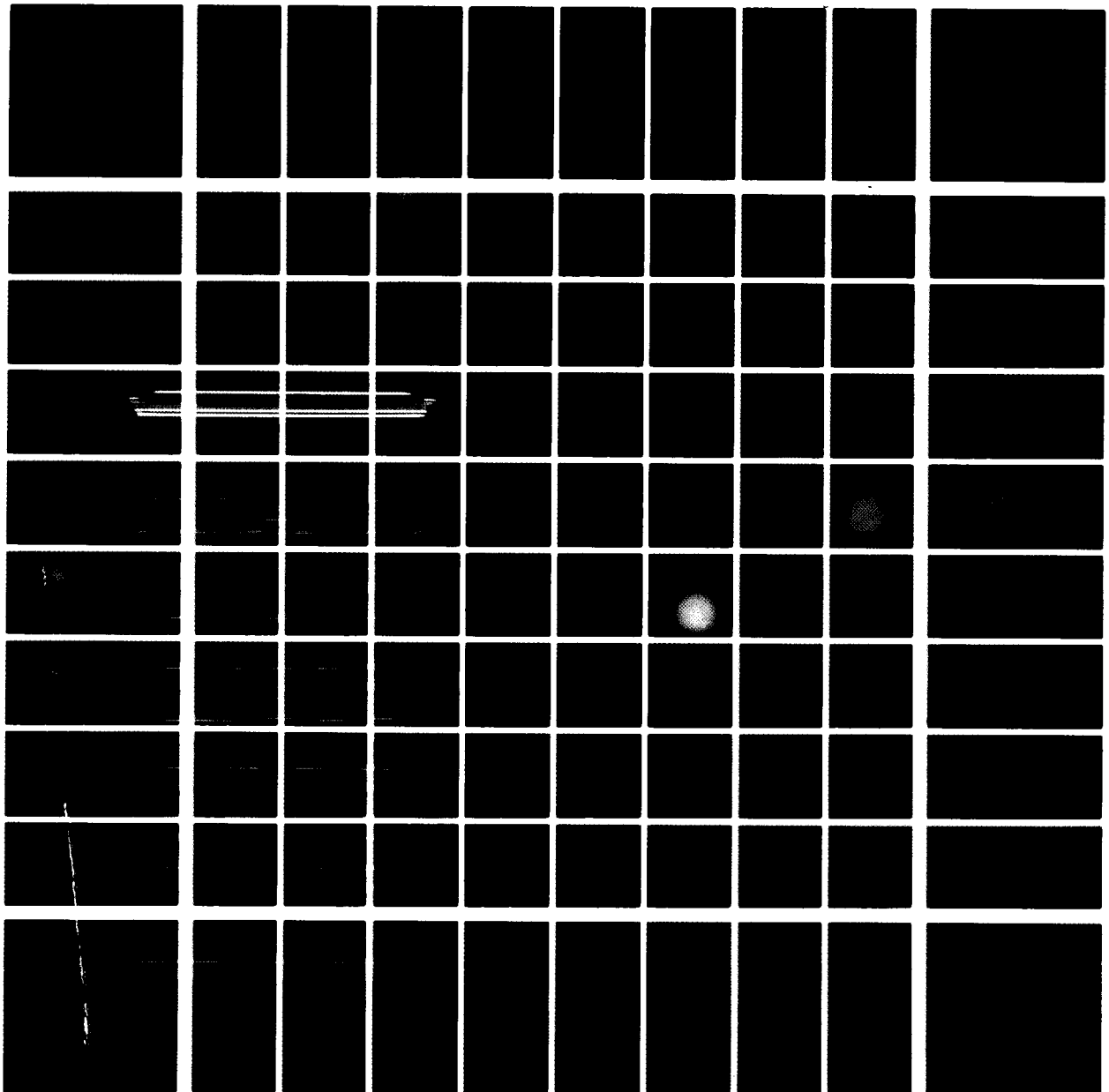


HEWLETT-PACKARD

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 and 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 in 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]** **[0]** **[.]** 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 **[ALPHA]** and the K key).
 - f. All operations requiring register addresses accept those addresses in these forms:

nn (a two-digit number)

IND nn (INDIRECT: **[IND]**, followed by a two-digit number)

X, Y, Z, T, or L (a STACK address: **[R/S]** followed by X, Y, Z, T, or L)

IND X, Y, Z, T or L (INDIRECT stack: **[IND]** **[R/S]** followed by X, Y, Z, T, or L)

Indirect addresses are specified by pressing **[IND]** and then the indirect address. Stack addresses are specified by pressing **[R/S]** followed by X, Y, Z, T, or L. Indirect stack addresses are specified by pressing **[IND]** **[R/S]** 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] [R/S] L
[ALPHA] R3= [ARCL] 03
[♦] AVIEW
[ALPHA]
[RTN]

```

Display

```

01 LBLT SAMPLE
02T THIS IS A
03T † SAMPLE
04 AVIEW
05 6
06 ENTER /
07 -2
08 /
09 ABS
10 STO IND L
11T 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

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...

3

[illegible]

Program Listings

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

Program Listings

```

98 RCL 01
99*LBL 08
100 RCL IND
10
101 X<>Y
102 X=Y?
103 GTO IND
10
104 ISG 10
105 GTO 08
106 GTO 20
107*LBL 02
108 "CHOMP"
109 AVIEW
110 RTN
111*LBL 03
112*LBL 04
113 "YYIIIEE
EE..."
114 AVIEW
115 RTN
116*LBL 05
117*LBL 06
118 "SNATCH"
119 AVIEW
120 20
121 XEQ 99
122 1
123 +
124 STO 01
125 "MOVED T
O "
126 ARCL 01
127 AVIEW
128 PSE
129 GTO 09
130*LBL 11
131 "GOT HIM
"
132 BEEP
133 AVIEW
134 RTN
135*LBL 14
136 CF 00
137 RCL 07
138 X=Y?
139 RTN
140 X<>Y
141 RCL 08
142 X=Y?
143 RTN
144 X<>Y

```

Check current
position for
dangers

Eaten by Wumpus

Fell into pit

Super bat move

Shoot Wumpus

Illegal cave
test

```

145 RCL 09
146 X=Y?
147 RTN
148 "ILLEGAL
CAVE"
149 AVIEW
150 SF 00
151 RTN
152*LBL 99
153 RCL 00
154 9821
155 *
156 .21137
157 +
158 FRC
159 STO 00
160 *
161 INT
162 RTN
163*LBL 50
164 CF 00
165 5
166 X<>Y
167 STO 10
168 X<=Y?
169 GTO 10
170 15
171 X<>Y
172 X<=Y?
173 GTO 11
174 2
175 *
176 25
177 -
178 STO 07
179 21
180 RCL 10
181 1
182 +
183 16
184 RDN
185 X=Y?
186 R↑
187 STO 08
188 15
189 RCL 10
190 1
191 -
192 X=Y?
193 20
194 X=0?
195 RDN

```

Random number
generator

Determine which
ring the cave is
in

Find adjacent
caves for outside
ring

Program Listings

196 STO 09		248 R↑	
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			
206 -		60	
207 X=0?			
208 5			
209 STO 08			
210 RCL 10			
211 2			
212 *			
213 4			
214 +			
215 STO 09			
216 RTN	----- Find adjacent caves for middle ring	70	
217♦LBL 11			
218 2			
219 /			
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		90	
237 +			
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 ⁷

DATA REGISTERS				STATUS							
00	Seed	50		SIZE	012	TOT. REG.	78	USER MODE			
	Hunter			ENG		FIX		SCI		ON	OFF
	Wumpus			DEG		RAD		GRAD			
	Pits			FLAGS							
	Pits										
05	S. bats	55		#	INIT	SET INDICATES		CLEAR INDICATES			
	S. bats			00	S/C	Used		Used			
	Adj. cave										
	Adj. cave										
	Adj. cave										
10	Used	60									
	Used										
	Used										
15		65									
20		70									
25		75									
30		80									
35		85									
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:

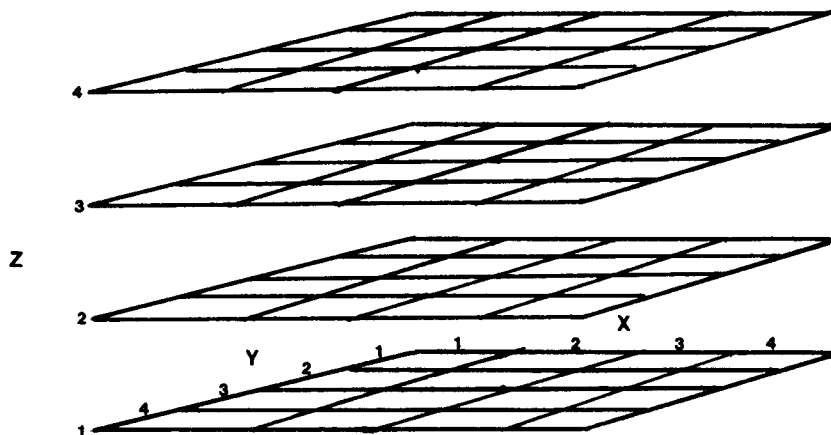
- 1) The board consists of 4 levels, each of which is 4 rows deep and 4 columns across, making a total of 64 squares on a 3 dimensional board.
- 2) Two players move alternately by placing a black or white marker on a square (or making an X or a 0 on a paper layout of the board). Once a move is made, 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.

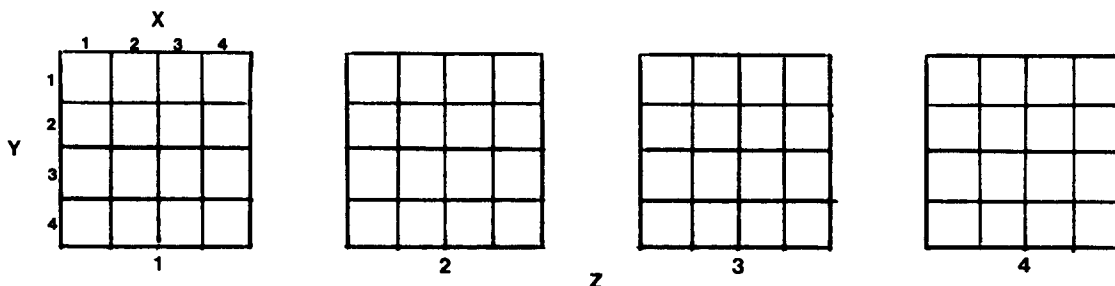
The 41C plays and remembers the game by dividing the board into its 16 component rows and storing an entire row in one register. The registers R_0 through R_{15} are reserved for the game board.

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] 3DTT [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: 114

MY MOVE: 434

334, I WIN

The boards look like:

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

1

			O	
		O		
	X			

2

X				O

3

O	X		X
			X
O	O	O	O

4

Z

Program Listings

11

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

Program Listings

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

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	Find move	348 INT	
302 RCL IND		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	Find empty space
311 GTO 02	-----	359 XEQ 19	
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	Find total moves	371 1 E8	-----
324♦LBL 07		372♦LBL 19	
325 RCL IND		373 STO 20	
25		374 RCL IND	Check/store move
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

15

397 *		51	
398 1			
399 ST+ IND			
25			
400 +			
401 X<>Y			
402 1			
403 +			
404 10	Place in XYZ		
405 *	form		
406 +		60	
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		
432 .END.	move		
40		90	
50		00	

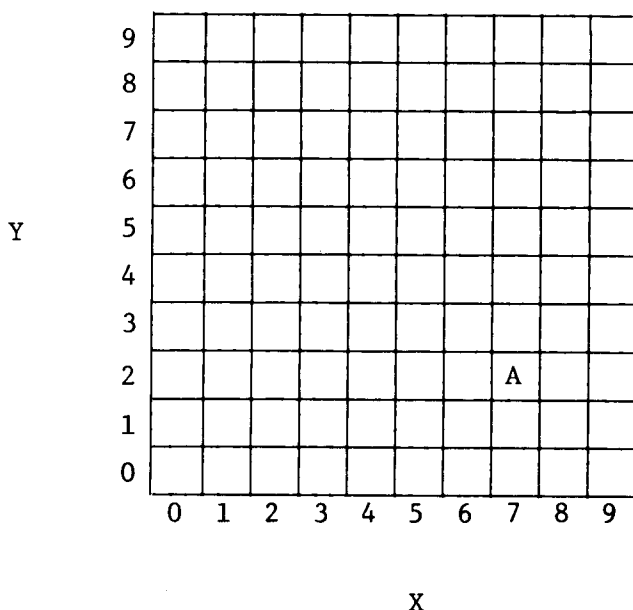
ROBOT TRAP

(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

⋮
⋮
⋮

19

[illegible]

Program Listings

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

Program Listings

21

97*LBL 16	Output loop	147 XEQ 96	
98 RCL IND		148 FS? 05	Android blunder
04		149 GTO 95	
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 "FAT "		155 RCL IND	Recall robot
105 ARCL X		03	
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	Input	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 -1.1	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	Robot blunder
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		
198 XEQ 96	android?		
199 FC? 05			
200 GTO 98			
201♦LBL 95			
202 "STOMP"	Android loses		
203 FS? 06		60	
204 "ZAP"			
205 AVIEW			
206 TONE 0			
207 "TOO BAD			
"			
208 AVIEW			
209 RTN			
210♦LBL 96			
211 SF 05	Android or robot		
212 SF 06	blunder routine	70	
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	

23

DATA REGISTERS				STATUS			
00	Seed	50		SIZE	6+	TOT. REG.	76+
	Used			ENG		FIX	
	Used			DEG		SCI	
	Used			RAD		GRAD	
	Used						
05	Used	55		FLAGS			
	Android			#	INIT S/C	SET INDICATES	CLEAR INDICATES
	Robot ₁			05		Used	
	:			06		Robot	F.F.
	:						
10	Robot _n	60					
	FF ₁						
	:						
	:						
	FF _{n+1}						
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:

•	7	•	8	•	9
	4		5		6
○	1	○	2	○	3

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:

Display:

[XEQ] [ALPHA] SIZE [ALPHA] 014

[XEQ] [ALPHA] MACHINE [ALPHA]

SEED?

.111111111 [R/S]

8 to 5

1.5 [R/S]

7 to 4

(A bad move; therefore, punish)

[E]

AAAAIIII...

5.8 [R/S]

YOU WIN

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

Keystrokes:

Display:

[A]

8 to 5

1.5 [R/S]

7 to 5

3.5 [R/S]

9 to 5

0 [R/S]

I WIN

Program Listings

27

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

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		175 XEQ 23	
6		176 1179.525	
139 XEQ 23		3	

Machine move
first boards
and responses

Program Listings

29

177 XEQ 23		226 RCL 04	Move loop
178 2286.747		227 1	
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	Check to see if	240 .5	
191 INT	correct board	241 X<=Y?	
192 RCL 10		242 GTO 04	
193 X*Y?		243 DSE 13	
194 RTN		244 GTO 02	
195 LASTX		245 RCL 07	
196 FRC		246 STO 05	
197 STO 06		247 STO 04	
198 RCL 13		248 FS? 08	
199 INT		249 CF 09	
200♦LBL 20	Move	250 FS? 09	
201 STO 08		251 RTN	
202 RCL 04		252♦LBL 25	-----
203 STO 09		253 "NO MOVE	No move possible
204 RCL 05		"	
205 STO 07		254 AVIEW	
206 2		255 PSE	
207 RCL 08		256♦LBL 26	
208 Y↑X		257 "YOU WIN	
209 STO 05		"	
210 3		258 AVIEW	
211 STO 13		259 STOP	
212 RCL 00		260♦LBL 04	-----
213 9821		261 RCL 04	
214 *		262 1	
215 .211327	Random number	263 X*Y?	Convert machines
216 +	generator	264 CF 08	move to board
217 FRC		265 -	notation
218 STO 00		266 2	
219 *		267 *	
220 1		268 10↑X	
221 +		269 RCL 06	
222 INT		270 *	
223 STO 04		271 FRC	
224♦LBL 02	-----	272 10	
225 3		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	
288 AVIEW		338 ENTER↑	Update board
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		362 3	
313 STO 06	Get mirror image move	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

31

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

SCATTER

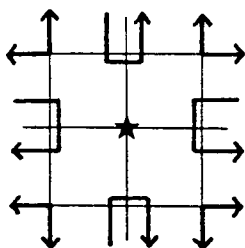
(Requires 1 Memory Module)

N atoms are randomly placed in a black box with dimensions 10x10. 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.

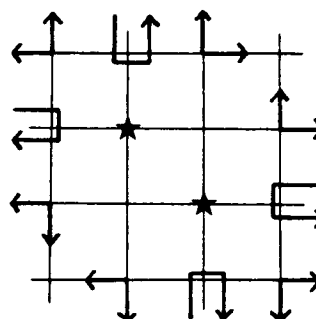
You select the value of N, and the machine places the N atoms randomly. You then fire particles from the edge: The machine tracks them and displays the output edge locations. At any time you can get the machine to confirm or reject any suspected atom location. If the guess is wrong, you are "penalized" 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.

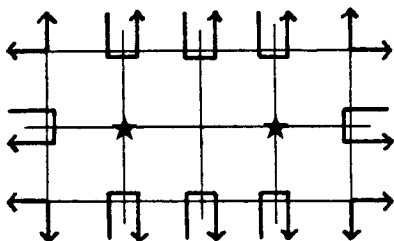
Diagrams



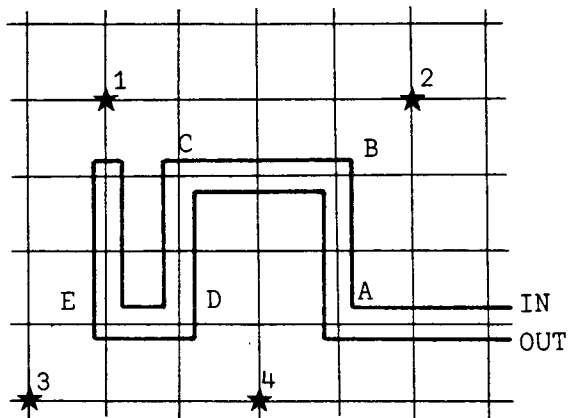
1. - SINGLE ATOM



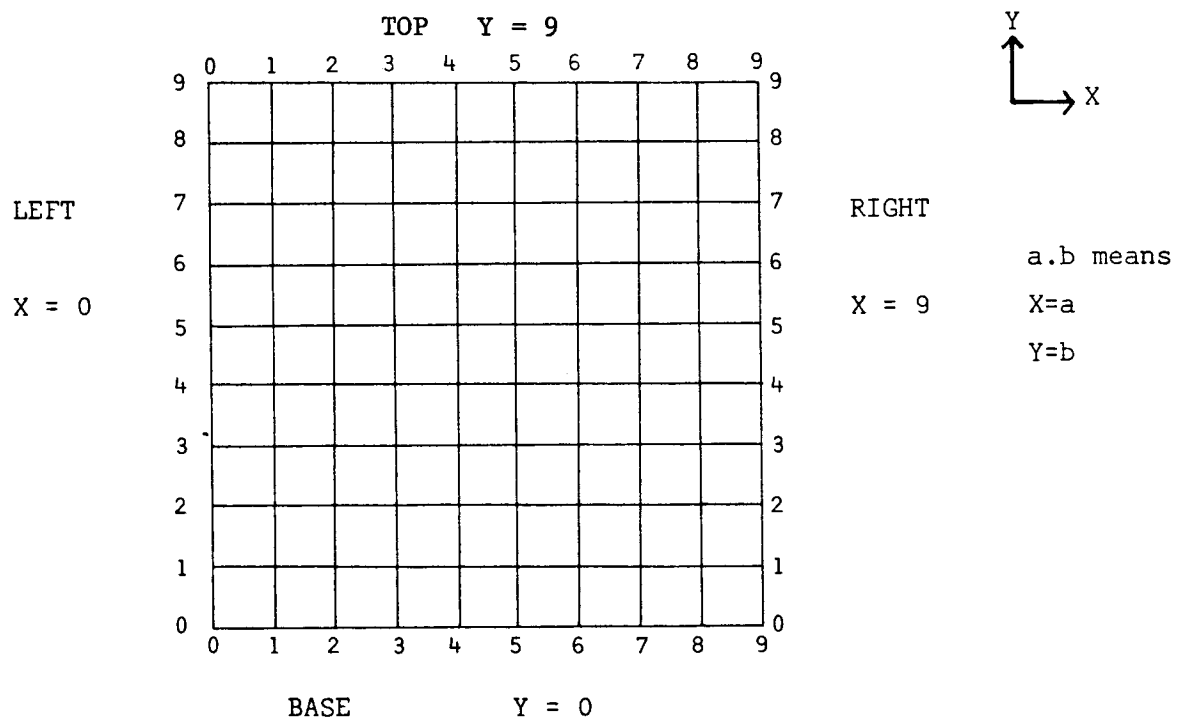
2. TWO ATOMS (1)



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

⋮

Program Listings

37

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

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		154 FS?C 03	
104 -	Test X distance	155 GTO 03	
105 X>0?		156 RCL 12	Deflection
106 GTO 09	No effect on	157 RCL 13	
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		170 STO 12	
120 STO 17	Flag 3 on for	171 FS? 00	
121 RCL 20	reflection	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		187 RCL IND	
137 RCL 13	Get exit	Y	
138 FS? 00	coordinates	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

39

199 PSE		51	
200 CLA			
201 ARCL 11			
202 "F PROBE			
S"			
203 AVIEW			
204 GTO 20			
205♦LBL C			
206 RCL 10			
207 FIX 1			
208 CF 28		60	
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	

Find and display
all positions

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

⋮

Program Listings

43

01♦LBL "FLI P"	Initialize	47 CHS	
02 CF 28		48 STO IND	
03 FIX 9		12	
04 CLRG		49 ST- 00	
05 9		50 DSE 12	
06 STO 12		51 DSE 12	
07♦LBL 00		52 RCL IND	
08 10		12	
09 RCL 12		53 CHS	
10 CHS		54 STO IND	
11 Y↑X		12	
12 STO IND		55 ST- 00	
12		56 GTO 01	
13 DSE 12		57♦LBL 09	
14 GTO 00		58 RCL 01	Right end
15 RCL 05		59 CHS	
16 STO 00		60 STO 01	
17 CHS		61 ST- 00	
18 STO 05		62 RCL 08	
19♦LBL 01		63 CHS	
20 RCL 00		64 STO 08	
21 .1111011		65 ST- 00	
11		66 GTO 01	
22 X<>Y		67♦LBL 04	
23 X=Y?		68 RCL 02	Left end
24 GTO 02		69 CHS	
25 RCL 10		70 STO 02	
26 +		71 ST- 00	
27 XEQ 05		72 RCL 09	
28♦LBL 03		73 CHS	
29 STO 12		74 STO 09	
30 STO 10		75 ST- 00	
31 XEQ 07		76 GTO 01	
32 ISG 11		77♦LBL 05	
33♦LBL 10		78 CF 22	
34 CHS		79 VIEW X	
35 STO IND	Update	80♦LBL 06	Input loop
12		81 PSE	
36 ST- 00		82 FS?C 22	
37 9		83 RTN	
38 RCL 12		84 GTO 06	
39 X=Y?		85♦LBL 07	Valid?
40 GTO 09		86 RCL IND	
41 1		12	
42 X=Y?		87 X<0?	
43 GTO 04		88 RTN	
44 ISG 12		89 RCL 00	
45♦LBL 10		90 VIEW X	
46 RCL IND		91 XEQ 06	
12		92 GTO 03	
		93♦LBL 02	

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 "F FLIPS		60	
"			
104 RVIEW			
105 .END.			
20		70	
30		80	
40		90	
50		00	

REGISTERS, STATUS, FLAGS, ASSIGNMENTS ⁴⁵

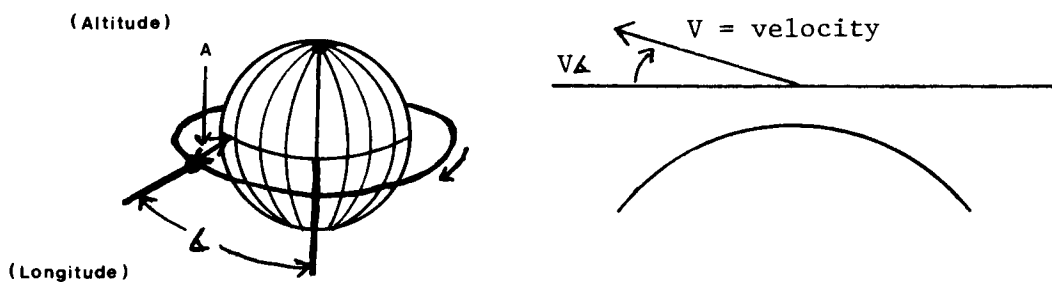
DATA REGISTERS			STATUS			
00	Current display	50	SIZE 013 TOT. REG. 48 USER MODE ENG FIX SCI ON OFF DEG RAD GRAD			
	Used		FLAGS # INIT S/C SET INDICATES CLEAR INDICATES			
	Used					
	Used					
	Used					
	Used					
05	Used	55				
	Used					
	Used					
	Used					
	Used					
10	Used	60				
	# of flips					
	Counter					
15		65				
20		70				
25		75				
30		80				
35		85				
			ASSIGNMENTS			
			FUNCTION KEY		FUNCTION KEY	
40		90				
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 $\pm 180^\circ$. 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)

 $\angle$ =0.00 (longitude)

V=1631.77 M/S (velocity)

 $V\angle$ =0.00 (angle from horizontal)

F=2,000.00 (fuel)

A=99,957.06 M

 $\angle$ =55.65

V=1,631.80 M/S

 $V\angle$ =0.00

F=2,000.00

A=99,908.77 M

 $\angle$ =55.95

V=941.88 M/S

 $V\angle$ =-0.59

F=1,310.00

A=78,392.28 M

 $\angle$ =61.89

V=974.77 M/S

 $V\angle$ =-12.14

F=1,310.00

⋮

Program Listings

49

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

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 /	g	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	Output
124 RCL 02		175 ADV	
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"	Altitude
133 RCL 03		184 AVIEW	
134 X<>Y		185 STOP	
135 /		186 "Δ="	
136 RCL 06		187 RCL 02	
137 +		188 1	
138 2		189 P-R	
139 *		190 R-P	
140 SQRT		191 ARCL Y	Longitude
141 STO 13		192 AVIEW	
142 RCL 01		193 STOP	
143 *		194 "V="	
144 RCL 07		195 ARCL 13	
145 X<>Y		196 "F M/S"	Velocity
146 /		197 AVIEW	
147 FIX 7		198 STOP	
148 RND		199 "VΔ="	
149 ACOS		200 ARCL 14	Angle from
150 RCL 07		201 AVIEW	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 +		80	
232 RCL 12			
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	Impact		
249 3			
250 *			
251 RCL 04			
252 X↑2		00	

[illegible]

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>g(f/s²)</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 ($g = 5.32 \text{ f/s}^2$).

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

55

[illegible]

Program Listings

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

Program Listings

57

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 *	ΔV
105 3		155 ST- 02	
106 STO 00		156 0	
107 RCL 04		157 STO 06	
108 *	ΔV	158 GT0 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 *		70	
121 +	ΔAltitude		
122 ST+ 06			
123 GT0 09			
124♦LBL 02			
125 "DEEP SP			
ACE..."	Find velocity		
126 RCL 01	calculation		
127 X≠0?			
128 GT0 03			
129 0	g=0	80	
130 STO 06			
131 RCL 02			
132 X<0?			
133 GT0 09			
134 PROMPT			
135♦LBL 03	g≠0		
136 RCL 01			
137 RCL 06			
138 *			
139 2		90	
140 *			
141 RCL 02			
142 X↑2			
143 +			
144 X<0?			
145 PROMPT			
146 SQRT			
147 RCL 02			
148 +			
149 RCL 01		00	

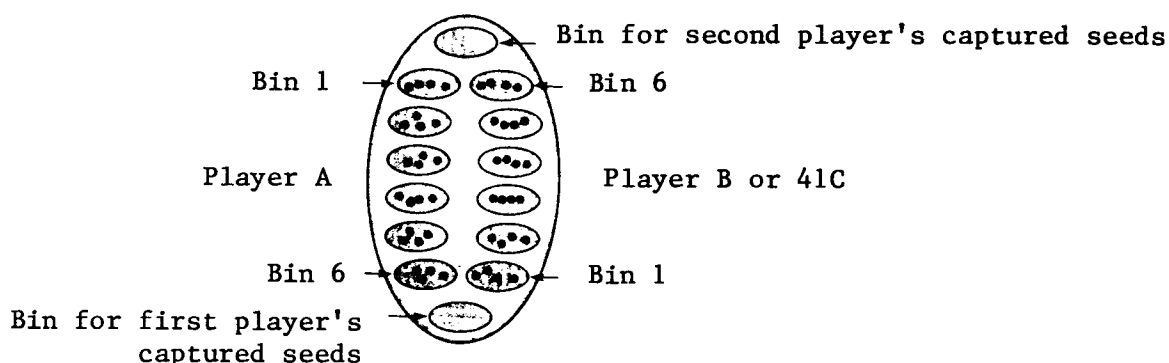
DATA REGISTERS				STATUS			
00	used	50		SIZE 007	TOT. REG. 40	USER MODE	
	g			ENG	FIX	SCI	ON OFF
	Vel.			DEG	RAD	GRAD	
	Fuel			FLAGS			
	Used						
05	Initial fuel	55		#	INIT S/C	SET INDICATES	CLEAR INDICATES
	Alt						
10		60					
15		65					
20		70					
25		75					
30		80					
35		85					
				ASSIGNMENTS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



Wari Board at start of 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
:
```


61

[illegible][illegible]

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<> L 47 7 48 X<=Y? </pre>	<pre> Initialize ----- 41C move Offense 41C pit # 41C counters Player pit # </pre>	<pre> 49 GTO 00 50 X<>Y 51 R↑ 52 CHS 53 RCL IND Y 54 X<=Y? 55 GTO 00 56 X<>Y 57 3 58 + 59 X<=Y? 60 GTO 00 61 SF 07 62 RCL Z 63 RCL 17 64 X>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>0? </pre>	<pre> Player counters Found an offensive move Defense Player's pit # Player's counters </pre>
--	--	---	---

Program Listings

63

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

65

DATA REGISTERS				STATUS			
00	Seed	50		SIZE 018 TOT. REG. 82 USER MODE			
	Bin A1			ENG _____ FIX _____ SCI _____ ON _____ OFF _____			
	A2			DEG _____ RAD _____ GRAD _____			
	A3						
	A4						
05	A5	55		FLAGS			
	A6			#	INIT S/C	SET INDICATES	CLEAR INDICATES
	Bin B1			05		A's move	B's move
	B2			06		Used	
	B3			07		Found one to attack	
10	B4	60					
	B5						
	B6						
	Score A						
	Score B						
15	Used	65					
	Used						
	Used						
20		70					
25		75					
30		80					
35		85					
40		90		ASSIGNMENTS			
				FUNCTION KEY		FUNCTION KEY	
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 π for the random number generation to duplicate this game.

Keystrokes:

[XEQ] [ALPHA] SIZE [ALPHA] 004

[////] [π] [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

67

				SIZE:004
STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
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^* (≤ 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	Set up tone counter
10 1.1111		56 STO 03	
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"		62 INT	
16 FIX 0	Set display format	63 VIEW X	
17 CF 29		64 TONE IND	
18 CF 06	Clear error flag	X	
19 FS?C 00		65 LASTX	Decrement tone count
20 SF 07		66 FRC	
21 XEQ "R"	Save status of F00	67 DSE 03	Repeat tone loop
22 FS?C 07		68 GTO 03	-----
23 SF 00		69 RTN	
24 "HOW MAN		70♦LBL "NO"	
Y?"		71 SF 06	
25 PROMPT		72 TONE 2	"Un-oh" sound
26 1 E3	Ask for maximum length	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	Increment counter
34 RCL 01	Get a sequence	80 1	
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	Increment counter	88 AVIEW	
43 AVIEW		89 RTN	
44 ISG 01	Repeat loop	90♦LBL "?"	Ask player to input numbers seen
45 GTO 10		91 "NUMBERS	
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	If correct, then done Otherwise, set error flag	51	
95 RCL 01			
96 INT			
97 10↑X			
98 *			
99 INT			
100 X=Y?			
101 RTN			
102 SF 05			
103 RTN		60	
104 .END.			
20		70	
30		80	
40		90	
50		00	

70 REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS			STATUS			
00	Random sequence length counter error counter tones counter	50	SIZE <u>004</u> TOT. REG. <u>40</u> USER MODE ENG <u> </u> FIX <u>0</u> SCI <u> </u> ON <u> </u> OFF <u> </u> DEG <u> </u> RAD <u> </u> GRAD <u> </u>			
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 26 tones no tones			
10		60				
15		65				
20		70				
25		75				
30		80				
35		85				
			ASSIGNMENTS			
			FUNCTION		KEY	
40		90				
45		95				

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:

**Aviation
Clinical Lab
Circuit Analysis
Financial Decisions
Mathematics**

**Structural Analysis
Surveying
Securities
Statistics
Stress Analysis
Games**

**Home Management
Machine Design
Navigation
Real Estate
Thermal and Transport Science**

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:

**Business Stat/Marketing/Sales
Home Construction Estimating
Lending, Saving and Leasing
Real Estate
Small Business
Geometry
High-Level Math
Test Statistics
Antennas
Chemical Engineering
Control Systems
Electrical Engineering
Fluid Dynamics and Hydraulics**

**Civil Engineering
Heating, Ventilating & Air Conditioning
Mechanical Engineering
Solar Engineering
Calendars
Cardiac/Pulmonary
Chemistry
Games
Optometry I (General)
Optometry II (Contact Lens)
Physics
Surveying**

* Some books require additional memory modules to accommodate all programs.

GAMES

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



Scan Copyright ©
The Museum of HP Calculators
www.hpmuseum.org

Original content used with permission.

Thank you for supporting the Museum of HP
Calculators by purchasing this Scan!

Please to not make copies of this scan or
make it available on file sharing services.