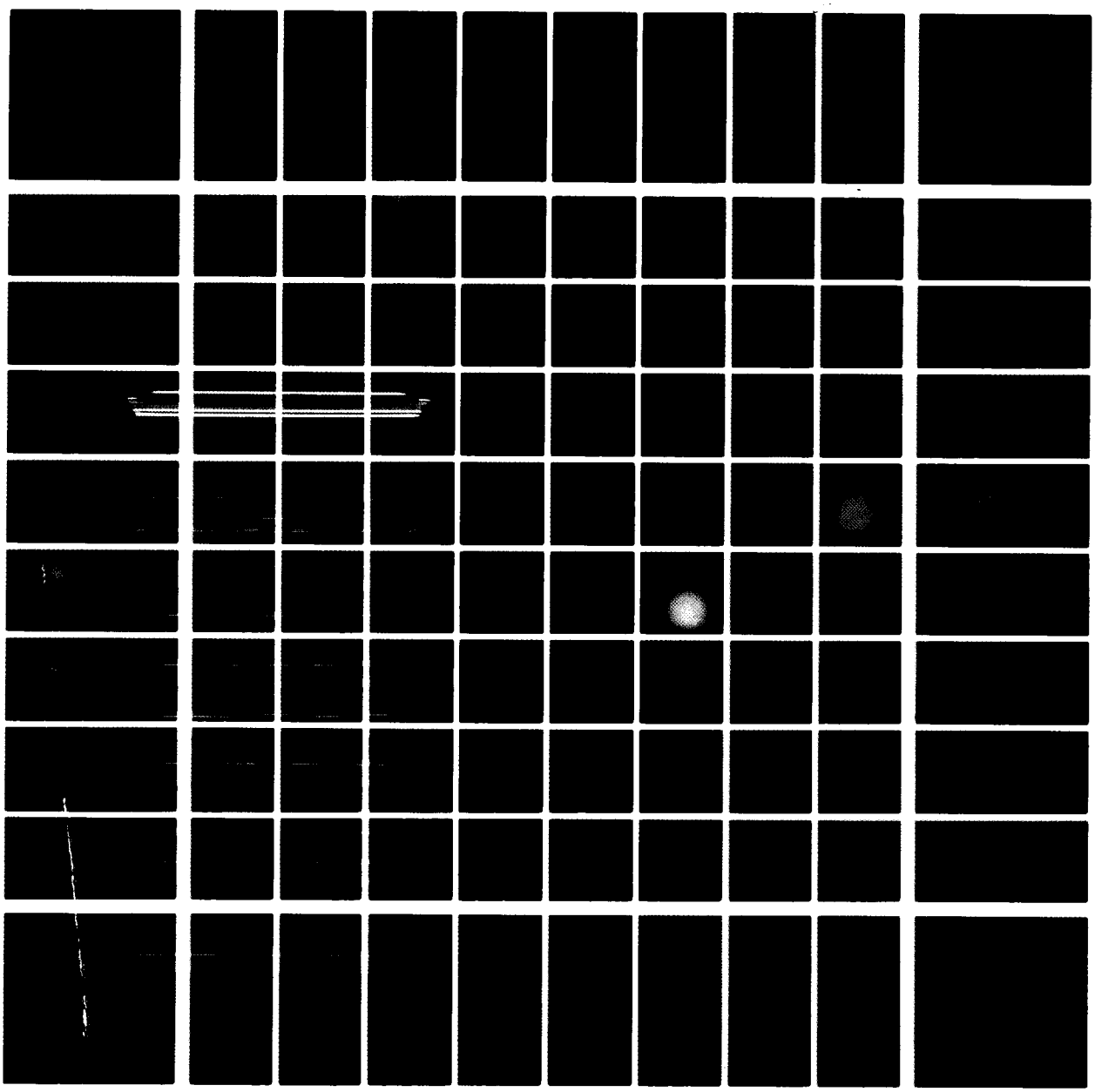
HEWLETT-PACKARD

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] [.] [.] 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 fy 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	Keystrokes	Display
01◆LBL "SAMPLE"	[■] [LBL] [ALPHA] SAMPLE [ALPHA]	*01 LBL"SAMPLE*
02 "THIS IS A "	[ALPHA] *THIS IS A* [ALPHA]	*02"THIS IS A*
03 "⊢SAMPLE"	[ALPHA] [■] [APPEND] SAMPLE	*03" ⊢ SAMPLE*
04 AVIEW	[■] AVIEW [ALPHA]	*04 AVIEW*
05 6	6	*05 6*
06 ENTER↑	[ENTER↑]	*06 ENTER ↑*
07 -2	2 [CHS]	*07 -2*
08 /	[÷]	*08 /*
09 ABS	[XEQ] [ALPHA] ABS [ALPHA]	*09 ABS*
10 STO IND L	[STO] [■] [.] L	*10 STO IND L*
11 "R3="	[ALPHA] R3= [■] [ARCL] 03	*11"R3=*
12 ARCL 03	[■] [AVIEW]	*12 ARCL 03*
13 AVIEW	[ALPHA]	*13 AVIEW*
14 RTN	[■] [RTN]	*14 RTN*

TABLE OF CONTENTS

* 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.

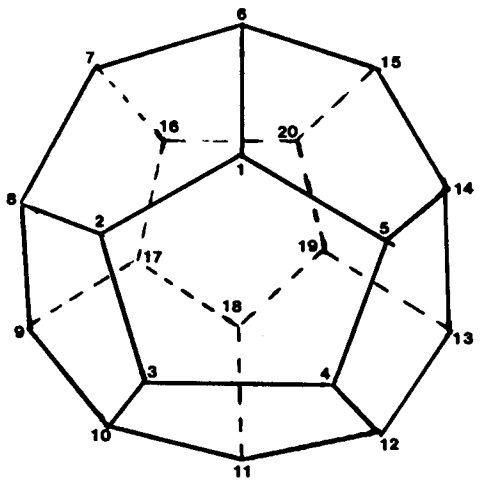
"FEEL A DRAFT" -- Pits are in adjacent cave(s).

"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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 012	
[XEQ] [ALPHA] WUMPUS [ALPHA]	SEED?
.1232123 [R/S]	HEAR SQUEAKS
[R/S]	6-1,7,15
1 [A]	SNATCH
[R/S]	MOVED TO 3
[R/S]	HEAR SQUEAKS
[R/S]	3-4,2,10
10 [A]	FEEL A DRAFT
[R/S]	10-3,11,9
11 [A]	SMELL WUMPUS
[R/S]	11-18,12,10
12 [B]	11-18,12,10
18 [A]	SMELL WUMPUS
[R/S]	18-11,19,17
19 [B]	18-11,19,17
17 [A]	SMELL WUMPUS
[R/S]	FEEL A DRAFT
[R/S]	17-9,18,16
9 [A]	YYIIIEEEE...

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

User Instructions

SIZE: 012

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load the program.			
2	Initialize.		[XEQ] WUMPUS	SEED?
3	Key in number between 0 and 1.	seed	[R/S]	(warnings)
				(caves)
4	To move:	cave #	[A]	(consequences)
				(warnings)
				(caves)
5	To shoot an arrow:	cave #	[B]	(consequences)
				(warnings)
				(caves)
6	To see warnings and caves again:		[C]	(warnings)
				(caves)
7	For another game:		[E]	(warnings)
				(caves)
8	Go to step 4 or 5.			

Program Listings

```
 01*LBL "WUM
PUS"
 02 0                    Initialize
 03 "SEED?"
 04 PROMPT
 05 STO 00
 06 SF 27
 07*LBL E
 08 FIX 0
 09 CF 29
 10 1.006
 11 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              Make sure it's
11                       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
 35 XEQ 50               surroundings
 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
 47 RCL 06
 48 XEQ 21               --------
 49*LBL 22
 50 CLA
 51 ARCL 01              Output current
 52 "⊢- "                position
 53 ARCL 07
 54 "⊢, "
 55 ARCL 08
 56 "⊢, "
 57 ARCL 09
 58 AVIEW
 59 RTN                  --------
 60*LBL 21
 61 RCL 07
 62 X<>Y                 Test surrounding
 63 X=Y?                 caves
 64 AVIEW
 65 RCL 08
 66 X<>Y
 67 X=Y?
 68 AVIEW
 69 RCL 09
 70 X=Y?
 71 AVIEW
 72 RTN                  --------
 73*LBL A
 74 XEQ 14
 75 FS? 00               Move
 76 GTO 22
 77 STO 01
 78 GTO 09
 79*LBL B                --------
 80 XEQ 14
 81 FS? 00               Shoot
 82 GTO 22
 83 RCL 02
 84 X=Y?                 Hit
 85 GTO 11
 86 XEQ 50
 87 3
 88 XEQ 99
 89 7
 90 +
 91 RCL IND
X
 92 STO 02               New Wumpus
 93 RCL 01               position
 94 XEQ 50
 95*LBL 09               --------
 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
197 RTN
198◆LBL 10
199 1
200 +
201 X>Y?
202 1
203 STO 07
204 RCL 10
205 1
206 -
207 X=0?
208 5
209 STO 08
210 RCL 10
211 2
212 *
213 4
214 +
215 STO 09
216 RTN
217◆LBL 11
218 2
219 /
220 FRC
221 X=0?
222 SF 00
223 25
224 RCL 10
225 FS? 00
226 4
227 FS? 00
228 CHS
229 CF 00
230 +
231 2
232 /
233 STO 07
234 16
235 RCL 10
236 1
237 +
238 X=Y?
239 6
240 STO 08
241 5
242 RCL 10
243 1
244 -
245 15
246 RDN
247 X=Y?
248 R↑
249 STO 09
250 RTN
251 .END.
```

Find adjacent caves for inside ring

Find adjacent caves for middle ring

60 70 80 90 00

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Seed	50	
	Hunter		
	Wumpus		
	Pits		
	Pits		
05	S. bats	55	
	S. bats		
	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	

STATUS

SIZE 012 TOT. REG. 78 USER MODE

ENG ____ FIX ____ SCI ____ ON ____ OFF ____

DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
00		Used	Used

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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:

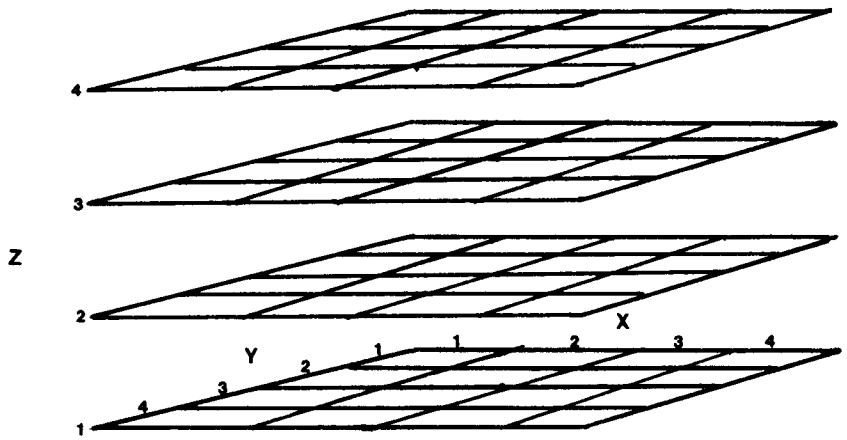
1) The board consists of 4 levels, each of which is 4 rows deep and 4 columns across, making a total of 64 squares on a 3 dimensional board.
2) Two players move alternately by placing a black or white marker on a square (or making an X or a 0 on a paper layout of the board). Once a move is made, the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

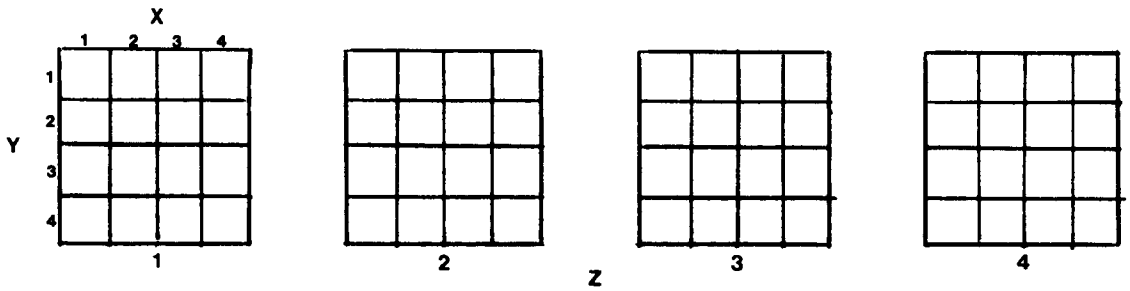
The 41C plays and remembers the game by dividing the board into its 16 component rows and storing an entire row in one register. The registers R_0 through R_{15} are reserved for the game board.

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:



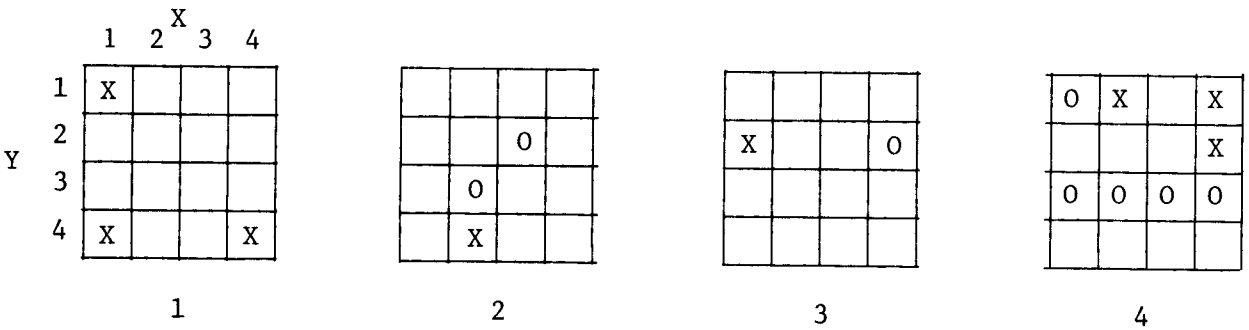
or



Example:

Keystrokes:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 026	
[XEQ] [ALPHA] 3DTTT [ALPHA]	READY
242 [R/S]	MY MOVE: 322
414 [R/S]	MY MOVE: 134
123 [R/S]	MY MOVE: 234
441 [R/S]	MY MOVE: 423
141 [R/S]	MY MOVE: 232
214 [R/S]	MY MOVE: 114
424 [R/S]	MY MOVE: 434
111 [R/S]	334, I WIN

The boards look like:



User Instructions

SIZE:026

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Initialize.		[XEQ] 3DTTT	READY
3	Key in your move.	xyz	[R/S]	MY MOVE:
	Repeat 3 until you win or lose.			

Program Listings


```
01◆LBL "3DTT"
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 -
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
```

Initialize test controls

```
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
```

Set test points

```
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
```

```
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 /
```

Testing cycle

Analyze sum

```
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
```

Program Listings

```
199 GTO 02
200◆LBL 01
201 5
202 /
203 "YOU WIN
"
204 X=Y?
205 PROMPT
206 FRC
207 X≠0?
208 RTN
209 LASTX
210◆LBL 02
211 RCL 17
212 X>Y?
213 RTN
214 X<>Y
215 SF 02
216 STO 17
217 FC? 00
218 SF 03
219 RCL 18
220 STO 16
221 RTN
222◆LBL 21
223 1
224 RCL 22
225 GTO 01
226◆LBL 22
227 4
228 RCL 21
229 GTO 01
230◆LBL 23
231 5
232 ENTER↑
233 0
234 GTO 01
235◆LBL 24
236 3
237 ENTER↑
238 GTO 01
239◆LBL 25
240 SF 01
241 0
242 RCL 23
243◆LBL 01
244 STO 25
245 RDN
246 STO 24
247 1 E2
248 ENTER↑
249 1 E8
```

Analyze player sum

Player wins

Test tactical status

Tactical play

```
250 FS? 00
251 X<>Y
252 FS? 01
253 STO 20
254 RCL 20
255◆LBL 05
256 XEQ 19
257 X<> 25
258 RCL 24
259 +
260 X<> 25
261 1 E-2
262 FS? 00
263 1/X
264 RCL 20
265 FS? 01
266 *
267 GTO 05
268◆LBL 20
269 16
270 STO 20
271 4
272 STO 25
273 1
274 STO 24
275 XEQ 07
276 2
277 XEQ 07
278 3
279 XEQ 07
280 0
281 XEQ 08
282 RCL 22
283 4
284 STO 24
285 *
286 STO 23
287 16
288 STO 20
289 1
290 XEQ 08
291 2
292 XEQ 08
293 3
294 XEQ 08
295 0
296 XEQ 08
297 RCL 22
298 RCL 23
299 +
300 STO 25
```

Find move

Strategic play

Program Listings

```
301◆LBL 09
302 RCL IND
25
303 RCL 24
304 X<=Y?
305 GTO 01
306 RCL 25
307 2
308 /
309 FRC
310 X=0?
311 GTO 02
312 GTO 03
313◆LBL 01
314 RCL 25
315 +
316 16
317 X>Y?
318 CLX
319 -
320 STO 25
321 GTO 09
322◆LBL 08
323 STO 25
324◆LBL 07
325 RCL IND
25
326 X<> 25
327 RCL 24
328 +
329 X<> 25
330 RCL IND
25
331 +
332 X<> 25
333 RCL 24
334 +
335 X<> 25
336 RCL IND
25
337 +
338 X<> 25
339 RCL 24
340 +
341 X<> 25
342 RCL IND
25
343 +
344 X<> 25
345 RCL 24
346 +
347 X<> 25
```

Find move

Reset reg. #

Find total moves

```
348 INT
349 RCL 20
350 X<=Y?
351 RTN
352 RDN
353 STO 20
354 RDN
355 STO 22
356 RTN
357◆LBL 02
358 1 E2
359 XEQ 19
360 1 E4
361 XEQ 19
362 1 E8
363 XEQ 19
364◆LBL 03
365 1 E6
366 XEQ 19
367 1 E4
368 XEQ 19
369 1 E2
370 XEQ 19
371 1 E8
372◆LBL 19
373 STO 20
374 RCL IND
25
375 *
376 INT
377 1 E2
378 /
379 FRC
380 X≠0?
381 RTN
382 RCL 20
383 1/X
384 ST+ IND
25
385 LOG
386 2
387 /
388 5
389 +
390 RCL 25
391 4
392 /
393 INT
394 LASTX
395 FRC
396 4
```

Find empty space

Check/store move

Program Listings


```
397 *
398 1
399 ST+ IND
25
400 +
401 X<>Y
402 1
403 +
404 10
405 *
406 +
407 10
408 *
409 +
410 CLA
411 ARCL X
412 1 E3
413 /
414 FS?C 02
415 GTO 01
416 STO 18
417 RCL 17
418 3
419 X>Y?
420 GTO 01
421 FC? 03
422 GTO 01
423 "⊢, I WI
N"
424 AVIEW
425 GTO A
426◆LBL 01
427 ASTO X
428 "MY MOVE
: "
429 ARCL X
430 AVIEW
431 GTO A
432 .END.
```

Place in XYZ form

41C wins

Get next player move

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Board	50	
	↓		
05		55	
10		60	
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	
40		90	
45		95	

STATUS

SIZE 026 TOT. REG. 133 USER MODE
ENG ____ FIX ____ SCI ____ ON ____ OFF ____
DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
00		Used	
01		Used	
02		Used	
03		Used	

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

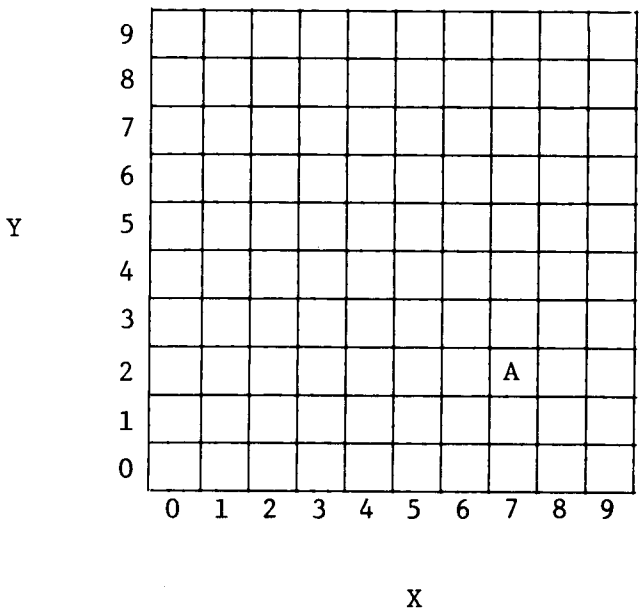
ROBOT TRAP

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

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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 014	
[XEQ] [ALPHA] RT [ALPHA]	SEED?
.12569 [R/S]	NO. OF ROBOTS
3 [R/S]	F.F. AT 7,8
[R/S]	F.F. AT 7,1
[R/S]	F.F. AT 3,6
[R/S]	F.F. AT 3,3
[R/S]	ROBOT AT 6,8
[R/S]	ROBOT AT 0,6
[R/S]	ROBOT AT 7,9
[R/S]	ANDROID: 6,1
3 [R/S]	STUMBLE
	ROBOT AT 7,7
[R/S]	ROBOT AT 1,5
[R/S]	ANDROID: 7,0
4 [R/S]	ROBOT AT 6,6
[R/S]	ROBOT AT 2,4
[R/S]	ANDROID: 6,0
⋮	⋮

User Instructions

SIZE: 6+

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Initialize.		[XEQ] RT	SEED?
3	Key in a seed between 1 and 0 for the			
	random number generator.	s	[R/S]	NO.OF ROBOTS
4	Key in the number of robots you wish.	n	[R/S]	SET SIZE(NNN)
5	Set size specified.		[XEQ] SIZE(NNN)	
	(If size was already set at or above		[R/S]	
	specified, the program will skip that			
	prompt.)			F.F. AT ( )*
6	Continue pressing [R/S] to see the rest of		[R/S]	⋮
	the force field and robot placements until		⋮	⋮
	the android placement appears.		[R/S]	ANDROID: ( )
7	Move the android (only when "ANDROID: ( )"			
	is in display).	direction	[R/S]	ROBOT AT ( )
8	Go to 6 to see the rest of the robot			
	placements and new android placement.			
9	For a new game		[E]	NO. OF ROBOTS
10	Go to step 4.			
*	Force field placements are only output at			
	the beginning of the game.			

Program Listings

```
01◆LBL "RT"
02 SF 27                    Initialize
03 "SEED?"
04 PROMPT
05 STO 00
06◆LBL E
07 "NO. OF
ROBOTS"
08 PROMPT
09 STO 03
10 2
11 *                        Calculate SIZE
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                  See if reg.
X                           exists
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
```

```
49◆LBL 00                   Get positions
50 RCL 04
51 100
52 XEQ 99
53 10
54 /
55◆LBL 11
56 RCL IND
Y
57 X=Y?                     Make sure no two
58 GTO 00                   are the same
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                   Output F.F's
73◆LBL 98                   once
74 CF 28
75 RCL 02
76 "ROBOT "
77 ASTO 03
78 CF 05
79 XEQ 10                   Output Robots
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                   Safe tune
91 TONE 9
92 TONE 7
93 TONE 9
94 RTN
95◆LBL 10
96 STO 04
```

Program Listings


```
 97◆LBL 16
 98 RCL IND
04
 99 X<0?
100 GTO 10
101 SF 05
102 CLA
103 ARCL 03
104 "HAT "
105 ARCL X
106 AVIEW
107 STOP
108◆LBL 10
109 ISG 04
110 GTO 16
111 RTN
112◆LBL 97
113 CF 22
114 STOP
115 FC? 22
116 GTO 98
117 GTO IND
X
118◆LBL 01
119 -1.1
120 GTO 10
121◆LBL 02
122 -.1
123 GTO 10
124◆LBL 03
125 .9
126 GTO 10
127◆LBL 04
128 -1
129 GTO 10
130◆LBL 05
131 0
132 GTO 10
133◆LBL 06
134 1
135 GTO 10
136◆LBL 07
137 -.9
138 GTO 10
139◆LBL 08
140 .1
141 GTO 10
142◆LBL 09
143 1.1
144◆LBL 10
145 ST+ 05
146 RCL 05
```

Output loop

Input

Change digit into ΔXY

Update android position

```
147 XEQ 96
148 FS? 05
149 GTO 95
150 RCL 02
151 STO 03
152◆LBL 14
153 RCL 05
154 INT
155 RCL IND
03
156 X<0?
157 GTO 10
158 INT
159 -
160 X≠0?
161 SIGN
162 RCL 05
163 FRC
164 RCL IND
03
165 FRC
166 -
167 X≠0?
168 SIGN
169 10
170 /
171 +
172 ST+ IND
03
173◆LBL 10
174 ISG 03
175 GTO 14
176 RCL 02
177 STO 03
178◆LBL 15
179 -1
180 X<> IND
03
181 X<0?
182 GTO 10
183 XEQ 96
184 FC? 05
185 GTO 10
186 -1
187 "BUMP"
188 FS? 06
189 "STUMBLE
"
190 AVIEW
191 TONE 9
192 CLD
193◆LBL 10
```

Android blunder

Move robots

Recall robot

Update robot

Robot blunder

Program Listings

```
194 STO IND
03
195 ISG 03
196 GTO 15
197 RCL 05
198 XEQ 96
199 FC? 05
200 GTO 98
201◆LBL 95
202 "STOMP"
203 FS? 06
204 "ZAP"
205 AVIEW
206 TONE 0
207 "TOO BAD
"
208 AVIEW
209 RTN
210◆LBL 96
211 SF 05
212 SF 06
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
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
232 CF 05
233 RTN
234◆LBL 99
235 RCL 00
236 9821
237 *
238 .211327
239 +
240 FRC
241 STO 00
242 *
243 INT
244 RTN
245 .END.
```

Robot get android?

Android loses

Android or robot blunder routine

Randon number generator

60 70 80 90 00

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Seed	50	
	Used		
	Used		
	Used		
	Used		
05	Used	55	
	Android		
	$Robot_1$		
	⋮		
10	$Robot_n$	60	
	FF_1		
	⋮		
	FF_{n+1}		
15		65	
20		70	
25		75	
30		80	
35		85	
40		90	
45		95	

STATUS

SIZE 6+ TOT. REG. 76+ USER MODE
ENG ____ FIX ____ SCI ____ ON ____ OFF ____
DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
05		Used	
06		Robot	F.F.

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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:

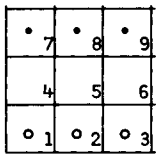
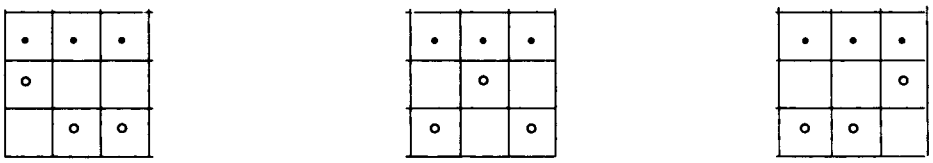


Figure 1. Starting position of pawns.

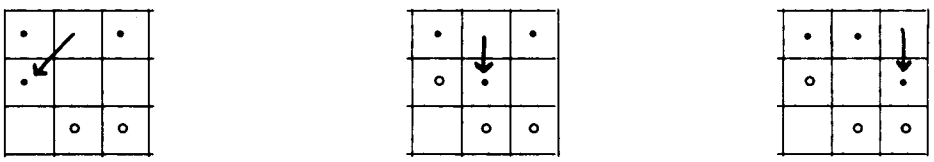
Note the resemblance between the 41C digit keys and the board numbering. The three allowed opening moves for the first player (in this example, white) are 1 to 4 (keyed as 1.4), 2 to 5 (2.5), and 3 to 6 (3.6).



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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 014	
[XEQ] [ALPHA] MACHINE [ALPHA]	SEED?
.1111111111 [R/S]	8 to 5
1.5 [R/S]	7 to 4
(A bad move; therefore, punish)	
[E]	AAAIIII...
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

User Instructions

SIZE: 014

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Key in program.			
2	Initialize type of game: human first,		[XEQ] HUMAN	SEED?
	machine first.		[XEQ] MACHINE	SEED?
3	Key in a seed for the random number			
	generator between 0 and 1. (Just hit [R/S]			
	if a seed has been previously entered.)	s	[R/S]	READY (or)
				( ) to ( )
4	Key in your move FROM.TO	F.T	[R/S]	( ) to ( )
	Repeat step 4 until game is over			
5	OPTIONAL: After the 41C displays its			
	move, punish it.		[E]	AAAIIII...
6	To signify that you can't move	0	[R/S]	I WIN
7	To start a new game with the same player			
	first and punishments remembered.		[A]	READY (or)
				( ) to ( )
	Go to step 4			
8	To start a new game with a different player			
	first and/or forgetting punishments go to			
	step 2.			

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
 98 1023.848
586
 99 XEQ 23
100 6758.515
286
101 XEQ 23
102 7163.958
6
103 XEQ 23
104 2720.414
2
105 XEQ 23
106 1131.847
5
107 XEQ 23
108 818.9596
109 XEQ 23
110 6650.959
6
111 XEQ 23
112 992.96
113 XEQ 23
114 677.4152
115 XEQ 23
116 369.75
117 XEQ 23
118 600.4186
119 XEQ 23
120 384.8463
121 XEQ 23
122 693.4152
123 XEQ 23
124 461.5263
125 XEQ 23
126 569.4195
96
127 XEQ 23
128 411.8452
129 XEQ 23
130 195.5286
131 XEQ 23
132 585.4175
133 XEQ 23
134 137.9563
135 XEQ 23
136 GTO 25
137◆LBL 22
138 9503.859
6
139 XEQ 23
```

Machine move
first boards
and responses

```
140 7341.748
586
141 XEQ 23
142 7449.747
5
143 XEQ 23
144 3237.848
562
145 XEQ 23
146 8957.745
1
147 XEQ 23
148 8849.959
6
149 SF 09
150 XEQ 23
151 CF 09
152 6.9
153 STO 13
154 8849.747
5
155 XEQ 23
156 6687.747
5
157 XEQ 23
158 855.7475
159 XEQ 23
160 1194.845
253
161 XEQ 23
162 2583.756
263
163 XEQ 23
164 6702.63
165 XEQ 23
166 1260.41
167 XEQ 23
168 1368.754
1
169 XEQ 23
170 6887.959
6
171 XEQ 23
172 2783.414
295
173 XEQ 23
174 6995.515
2
175 XEQ 23
176 1179.525
3
```

Program Listings


```
177 XEQ 23
178 2286.747
5
179 XEQ 23
180 2270.959
6
181 XEQ 23
182 2594.96
183 XEQ 23
184 621.4163
185 XEQ 23
186 432.52
187 XEQ 23
188 GTO 25
189◆LBL 23
190 ISG 13
191 INT
192 RCL 10
193 X≠Y?
194 RTN
195 LASTX
196 FRC
197 STO 06
198 RCL 13
199 INT
200◆LBL 20
201 STO 08
202 RCL 04
203 STO 09
204 RCL 05
205 STO 07
206 2
207 RCL 08
208 Y↑X
209 STO 05
210 3
211 STO 13
212 RCL 00
213 9821
214 *
215 .211327
216 +
217 FRC
218 STO 00
219 *
220 1
221 +
222 INT
223 STO 04
224◆LBL 02
225 3
```

Check to see if correct board

Move

Random number generator

```
226 RCL 04
227 1
228 +
229 X>Y?
230 1
231 STO 04
232 X<> 13
233 RCL IND
13
234 X<>Y
235 X<> 13
236 RDN
237 RCL 05
238 /
239 FRC
240 .5
241 X<=Y?
242 GTO 04
243 DSE 13
244 GTO 02
245 RCL 07
246 STO 05
247 STO 04
248 FS? 08
249 CF 09
250 FS? 09
251 RTN
252◆LBL 25
253 "NO MOVE
"
254 AVIEW
255 PSE
256◆LBL 26
257 "YOU WIN
"
258 AVIEW
259 STOP
260◆LBL 04
261 RCL 04
262 1
263 X≠Y?
264 CF 08
265 -
266 2
267 *
268 10↑X
269 RCL 06
270 *
271 FRC
272 10
273 *
```

Move loop

No move possible

Convert machines move to board notation

Program Listings

```
274 INT
275 STO 11
276 LASTX
277 FRC
278 10
279 *
280 INT
281 STO 12
282 SF 06
283 XEQ 21
284 3
285 RCL 12
286 X>Y?
287 GTO 00
288 AVIEW
289 BEEP
290◆LBL 00
291 RCL 11
292 RCL 12
293 10
294 /
295 +
296 XEQ 50
297 FIX 0
298 CF 29
299 CLA
300 INT
301 ARCL X
302 "⊢ TO "
303 LASTX
304 FRC
305 10
306 *
307 ARCL X
308 AVIEW
309 GTO 30
310◆LBL 50
311 FC? 07
312 RTN
313 STO 06
314 INT
315 1
316 -
317 3
318 /
319 INT
320 5
321 +
322 XEQ IND
X
323 RCL 06
```

41C win

41C move display and board update

Get mirror image move

```
324 -
325 RTN
326◆LBL 05
327 5
328 RTN
329◆LBL 06
330 10.4
331 RTN
332◆LBL 07
333 17
334 RTN
335◆LBL 21
336 RCL 10
337 3
338 ENTER↑
339 10
340 ENTER↑
341 RCL 12
342 -
343 Y↑X
344 /
345 FRC
346 3
347 *
348 INT
349 CHS
350 1
351 FS? 06
352 ST+ X
353 +
354 3
355 ENTER↑
356 9
357 ENTER↑
358 RCL 12
359 -
360 Y↑X
361 *
362 3
363 ENTER↑
364 9
365 RCL 11
366 -
367 Y↑X
368 1
369 FS? 06
370 ST+ X
371 *
372 -
373 ST+ 10
374 RTN
```

Update board

Program Listings


```
375*LBL E
376 RCL 04
377 X<> 13
378 RCL 05
379 2
380 /
381 ST- IND
13
382 RDN
383 X<> 13
384 "AAAIIII
..."
385 AVIEW
386 GTO 30
387 .END.
```

Punish

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	seed	50	
	Possible moves		
	Possible moves		
	Possible moves		
	Trial move		
05	Used	55	
	Used		
	Used		
	Current move		
	Last move		
10	Board	60	
	From		
	To		
	counter		
15		65	
20		70	
25		75	
30		80	
35		85	
40		90	
45		95	

STATUS

SIZE 014 TOT. REG. 167 USER MODE

ENG ___ FIX ___ SCI ___ ON ___ OFF ___

DEG ___ RAD ___ GRAD ___

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
05		41C first	Human first
06		41C's move	human's move
07		mirror image	normal game
08		first move	not first move
09		Used	Used

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

SCATTER

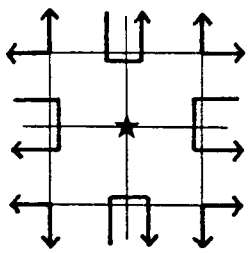
(Requires 1 Memory Module)

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.

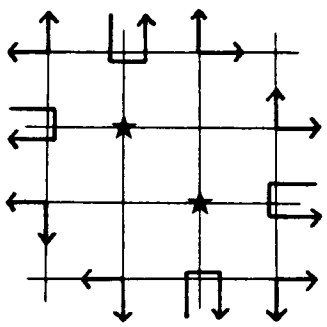
You select the value of N, and the machine places the N atoms randomly. You then fire particles from the edge: The machine tracks them and displays the output edge locations. At any time you can get the machine to confirm or reject any suspected atom location. If the guess is wrong, you are "penalized"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

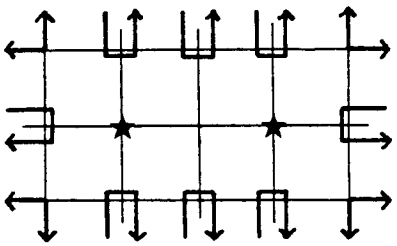
Diagrams



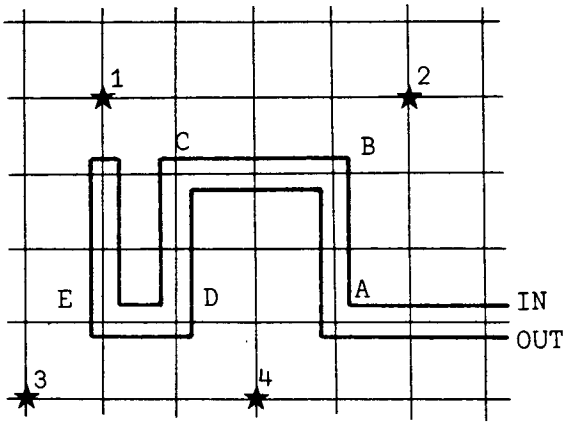
1. - SINGLE ATOM



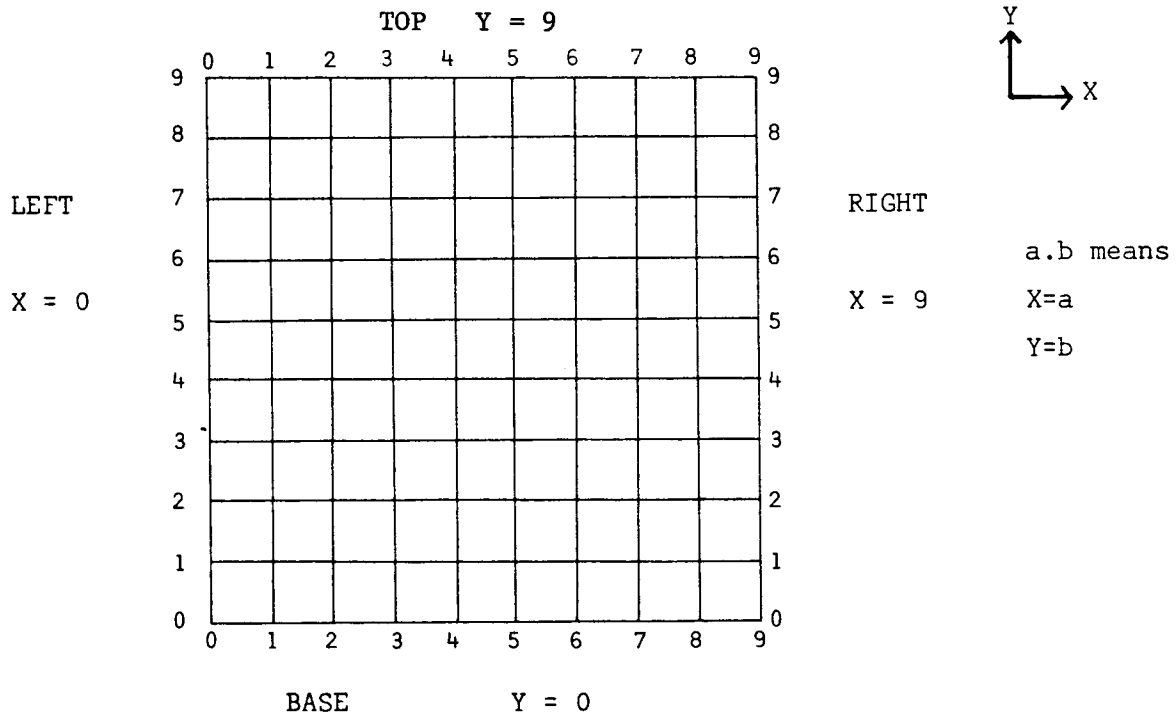
2. TWO ATOMS (1)



3. TWO ATOMS (2)



4. COMPLEX REFLECTION



5. BOX NUMBERING AND AXES

Example:

Set up and find 4 atoms.

Keystrokes:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 022	
[XEQ] [ALPHA] SCATTER [ALPHA]	SEED?
.191062 [R/S]	NO. OF ATOMS?
4 [R/S]	READY
2.0 [R/S]	0,2
4.0 [R/S]	9,2
3.3 [A]	YES
	2 PROBES
6.9 [R/S]	6,9
6.8 [A]	NO
	8 PROBES
0.8 [R/S]	9,8
⋮	⋮

User Instructions

SIZE: 022

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Initialize.		[XEQ] SCATTER	SEED?
3	Key in a seed for the random number			
	generator.	s	[R/S]	NO. OF ATOMS?
4	Key in number of atoms to be placed.	N	[R/S]	READY
5	Key in an entrance point.	x.y	[R/S]	(x).(y)
6	To guess a position key one in.	x.y	[A]	YES (or) NO
				( ) PROBES
	Continue with steps 5 and 6 as desired.			
7	To start a new game go to step 4.		[E]	NO. OF ATOMS?
8	If at any time you give up you can find			
	all positions.		[C]	(x).(y)
				(x).(y)
				⋮
				⋮

Program Listings

```
 01◆LBL "SCA
TTER"
 02 SF 27
 03 CF 29
 04 SF 28
 05 FIX 0
 06 "SEED?"
 07 PROMPT
 08 STO 00
```

Initialize

```
 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

```
 20◆LBL 00
 21 XEQ 10
 22 XEQ 10
 23 10
 24 /
 25 +
 26 RCL 12
 27 INT
 28 X<>Y
 29 STO 09
 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
```

Place atoms

Test to see if already filled

```
 44◆LBL 10
 45 RCL 00
 46 9821
 47 *
 48 .211327
 49 +
 50 FRC
 51 STO 00
 52 8
 53 *
 54 1
 55 +
 56 INT
 57 RTN
```

Get a coordinate

```
 58◆LBL 20
 59 STOP
 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
 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
 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 entry point

Select which side

Get an atom

Decompose into X and Y

Get X,Y distances of probe position from atom

Program Listings

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

Program Listings


```
199 PSE
200 CLA
201 ARCL 11
202 "⊢ PROBE
S"
203 AVIEW
204 GTO 20
205◆LBL C
206 RCL 10
207 FIX 1
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
218 GTO 20
219 .END.
```

Find and display all positions

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Seed	50	
	Atom locations		
05		55	
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	
45		95	

STATUS

SIZE 022 TOT. REG. 75 USER MODE
ENG ____ FIX ____ SCI ____ ON ____ OFF ____
DEG ____ RAD ____ GRAD ____

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

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 013	
[XEQ] [ALPHA] FLIP [ALPHA]	0,000010000
5	5,000Q1000
6	6,000110100
5	5,000001100
⋮	⋮

User Instructions

SIZE:013

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Initialize.		[XEQ] FLIP	0,000010000
3	Key in position to flip.	x		(x), ( )
	Repeat step 3 until successful.			

Program Listings


```
 01◆LBL "FLI
P"                    Initialize
 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 .1111011
11
 22 X<>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           Update
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
 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           Right end
 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           Left end
 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           Input loop
 81 PSE
 82 FS?C 22
 83 RTN
 84 GTO 06
 85◆LBL 07           Valid?
 86 RCL IND
12
 87 X<0?
 88 RTN
 89 RCL 00
 90 VIEW X
 91 XEQ 06
 92 GTO 03
 93◆LBL 02
```

Program Listings

```
 94 RCL 10
 95 +
 96 VIEW X
 97 PSE
 98 CLA
 99 FIX 0
100 CF 29
101 SF 28
102 ARCL 11
103 "⊢ FLIPS
"
104 AVIEW
105 .END.
```

Game over

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Current display	50	
	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	
40		90	
45		95	

STATUS

SIZE 013 TOT. REG. 48 USER MODE
ENG ____ FIX ____ SCI ____ ON ____ OFF ____
DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES

ASSIGNMENTS

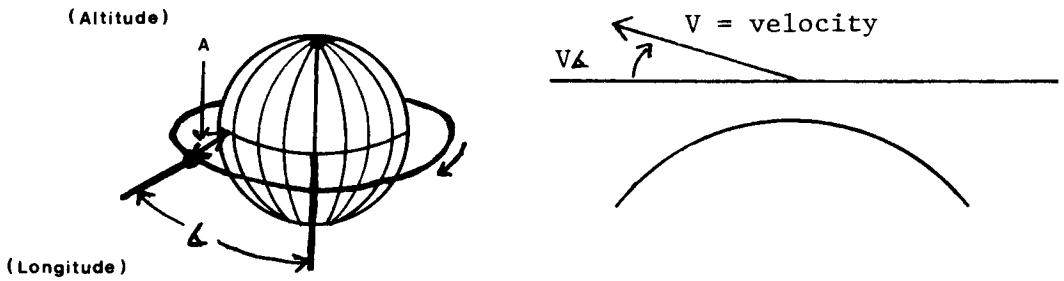
FUNCTION	KEY	FUNCTION	KEY

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 <u>not</u> 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. <u>No</u> 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∠) is given from horizontal, with 0° being forward and 90° straight up.
Thrust angles also follow V∠ conventions. 180° is a retrofire.

Example:

Keystrokes:	Display:	
[XEQ] [ALPHA] SIZE [ALPHA] 015		
[XEQ] [ALPHA] ORBIT [ALPHA]	A=100000.00 M	(altitude)
[R/S]	∠=0.00	(longitude)
[R/S]	V=1631.77 M/S	(velocity)
[R/S]	V∠=0.00	(angle from horizontal)
[R/S]	F=2,000.00	(fuel)
1000 [A]	A=99,957.06 M	
[R/S]	∠=55.65	
[R/S]	V=1,631.80 M/S	
[R/S]	V∠=0.00	
[R/S]	F=2,000.00	
For 10 seconds apply 7 gravities as retrofire		
69 [ENTER↑] 10 [X]		
180 [ENTER↑]		
10 [B]	A=99,908.77 M	
[R/S]	∠=55.95	
[R/S]	V=941.88 M/S	
[R/S]	V∠=-0.59	
[R/S]	F=1,310.00	
200 [A]	A=78,392.28 M	
[R/S]	∠=61.89	
[R/S]	V=974.77 M/S	
[R/S]	V∠=-12.14	
[R/S]	F=1,310.00	
⋮	⋮	

User Instructions

SIZE:015

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Initialize.		[XEQ] ORBIT	
3	*Mission status: altitude			A=
	longitude		[R/S]	∠=
	velocity		[R/S]	V=
	angle of flight		[R/S]	V∠=
	fuel remaining		[R/S]	F=
4	To free fall: key in number of seconds.	n	[A]	
	Go to step 3 for outputs.			
5	To fire rockets: key in total change in V	dV (m/s)	[ENTER↑]	
	key in angle of thrust	θ (deg)	[ENTER↑]	
	key in number of seconds			
	for total burn	n (sec)	[B]	
	Go to step 3 for outputs.			
	When A=0.00, you are down.			
*	Continuing [R/S] will repeat status.			

Program Listings


```
 01◆LBL "ORB
IT"                    Initialize
 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                  Thrust
 20 *
 21 R↑
 22 RCL 00
 23 X<=Y?
 24 X<>Y                No greater than
 25 RDN                 7 gees
 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                 Compute new
 41 RCL 05              orbit
 42 X↑2
 43 +
 44 2
 45 /
 46 RCL 03
 47 RCL 01
 48 /
 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                   Free fall
 95 ENTER↑
 96 ENTER↑
 97 RCL 05
 98 9
 99 X>Y?
```

Program Listings

```
100 GTO 21
101 RDN
102 RCL 12
103 *
104 RCL 03
105 RCL 01
106 X↑2
107 /
108 RCL 12
109 *
110 2
111 /
112 RCL 04
113 X<>Y
114 -
115 RCL 12
116 *
117 RCL 01
118 +
119 R-P
120 RDN
121 ST+ 02
122 RCL 08
123 RCL 09
124 RCL 02
125 RCL 10
126 -
127 COS
128 *
129 1
130 +
131 /
132 STO 01
133 RCL 03
134 X<>Y
135 /
136 RCL 06
137 +
138 2
139 *
140 SQRT
141 STO 13
142 RCL 01
143 *
144 RCL 07
145 X<>Y
146 /
147 FIX 7
148 RND
149 ACOS
150 RCL 07
```

g

```
151 RCL 02
152 RCL 10
153 -
154 SIN
155 *
156 X<0?
157 SF 05
158 RDN
159 FS?C 05
160 CHS
161◆LBL 20
162 STO 14
163 RCL 13
164 P-R
165 STO 05
166 RDN
167 STO 04
168 RCL 01
169 RCL 11
170 -
171 X<0?
172 GTO 22
173◆LBL 10
174 FIX 2
175 ADV
176 "A="
177 RCL 01
178 RCL 11
179 -
180 X<0?
181 CLX
182 ARCL X
183 "⊢ M"
184 AVIEW
185 STOP
186 "∠="
187 RCL 02
188 1
189 P-R
190 R-P
191 ARCL Y
192 AVIEW
193 STOP
194 "V="
195 ARCL 13
196 "⊢ M/S"
197 AVIEW
198 STOP
199 "V∠="
200 ARCL 14
201 AVIEW
```

Output

Altitude

Longitude

Velocity

Angle from horizon

Program Listings

```
202 STOP
203 "F="
204 ARCL 00
205 AVIEW
206 STOP
207 GTO 10
208◆LBL 21
209 RDN
210 RDN
211 2
212 /
213 CHS
214 RCL 05
215 +
216 RCL 12
217 *
218 X<>Y
219 RCL 03
220 RCL 01
221 X↑2
222 /
223 RCL 12
224 *
225 -
226 ST+ 04
227 2
228 /
229 CHS
230 RCL 04
231 +
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
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
```

Fuel

Vector sums

```
253 +
254 ABS
255 SQRT
256 CHS
257 STO 04
258 GTO 00
259 .END.
```

60

70

80

90

Impact

00

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Fuel	50	
	Ri		
	θi		
	Gm		
	Vr		
05	Ve	55	
	E		
	1		
	Ko		
	e		
10	θ'	60	
	surface		
	t		
	V		
	θ_v		
15		65	
20		70	
25		75	
30		80	
35		85	
40		90	
45		95	

STATUS

SIZE 015 TOT. REG. 66 USER MODE
ENG ___ FIX ___ SCI ___ ON ___ OFF ___
DEG ___ RAD ___ GRAD ___

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
05		Used	Used

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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:

Body	$g(f/s^2)$
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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 007	
5.32 [XEQ] [ALPHA] GRAVITY [ALPHA]	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:	Display:
	V=-516 F/S
	A=3476 F
	THREE...
	TWO...
	ONE...
	ZERO...
20	FUEL=5436
	V=-512 F/S
	A=1934 F
	THREE...
	TWO...
	ONE...
	⋮

User Instructions

SIZE:007

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load program.			
2	Key in gravity.	g(f/s^2)	[XEQ] GRAVITY	G=
3				FUEL= center
				*V=
				A=
				THREE...
				TWO...
				ONE...
				ZERO...
4	You have one second during the "ZERO..."			
	prompt to key in a fuel burn.	burn		
5	Go to 3 for next status.			
6	For a new game with the same g		[A]	G=
	go to step 3.			
7	For a new game with a different g go to			
	step 2.			
*	When you see "VF= ", this is your			
	landing or crash velocity.			

Program Listings

```
 01◆LBL "GRA
VITY"
 02 SF 27
 03 STO 01
 04 ABS
 05 800
 06 *
 07 1200
 08 +
 09 STO 05
 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
 24◆LBL 09
 25 "FUEL="
 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 "⊢="
 37 ARCL Z
 38 "⊢ F/S"
 39 AVIEW
 40 PSE
 41 X>Y?
 42 RTN
 43 "A="
 44 ARCL 06
 45 "⊢ F"
 46 AVIEW
 47 PSE
 48 RCL 03
 49 X=0?
 50 GTO 02
```

Initialize

Set up initial conditions

Display status

```
 51 "THREE..
."
 52 AVIEW
 53 PSE
 54 "TWO..."
 55 AVIEW
 56 PSE
 57 "ONE..."
 58 AVIEW
 59 PSE
 60 CLX
 61 "ZERO...
"
 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
 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
```

Countdown

Calculate acceleration

Calculate time to impact

If greater than 3 seconds use 3

Program Listings

```
100 X<>Y
101 X>Y?
102 RDN
103◆LBL 01
104 FS?C 00
105 3
106 STO 00
107 RCL 04
108 *
109 RCL 02
110 +
111 X<> 02
112 RCL 00
113 X↑2
114 RCL 04
115 *
116 2
117 /
118 X<>Y
119 RCL 00
120 *
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
130 STO 06
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
```

ΔV

ΔAltitude

Find velocity calculation

g=0

g≠0

```
150 /
151 X<0?
152 PROMPT
153 RCL 01
154 *
155 ST- 02
156 0
157 STO 06
158 GTO 09
159 .END.
```

t

ΔV

70

80

90

00

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	used	50	
	g		
	Vel.		
	Fuel		
	Used		
05	Initial fuel	55	
	Alt		
10		60	
15		65	
20		70	
25		75	
30		80	
35		85	
40		90	
45		95	

STATUS

SIZE 007 TOT. REG. 40 USER MODE

ENG ____ FIX ____ SCI ____ ON ____ OFF ____

DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES

ASSIGNMENTS

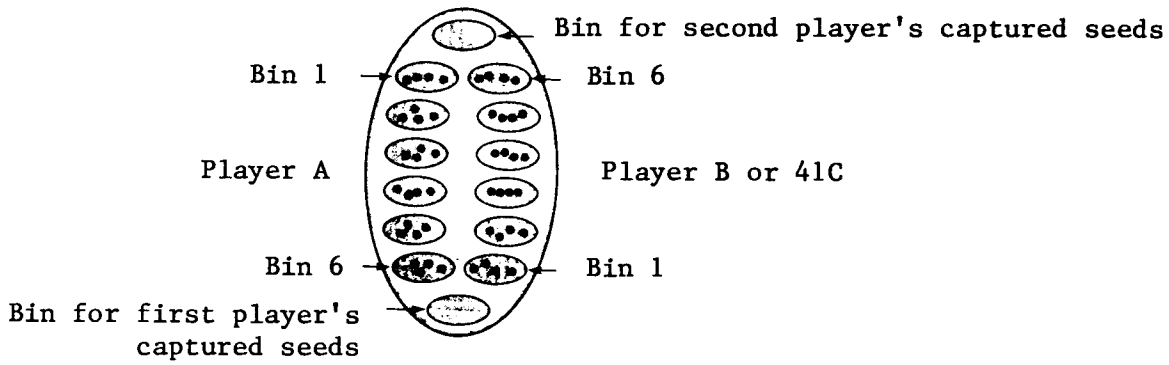
FUNCTION	KEY	FUNCTION	KEY

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:	Display:
[XEQ] [ALPHA] SIZE [ALPHA] 018	
[XEQ] [ALPHA] WARI [ALPHA]	SEED?
.9977663333 [R/S]	A 4,4,4,4,4,4
[R/S]	B 4,4,4,4,4,4
[R/S]	A=0, B=0
[C]	A 5,5,4,4,4,4
[R/S]	B 4,4,4,0,5,5
6 [A]	A 5,5,4,4,4,0
[R/S]	B 5,5,5,1,5,5
[C]	A 6,6,4,4,4,0
[R/S]	B 5,5,0,2,6,6
2 [A]	A 6,0,5,5,5,1
[R/S]	B 6,6,0,2,6,6,
[C]	A 7,1,6,6,6,0
[R/S]	B 6,6,0,2,6,0
[R/S]	A=0, B=2
⋮	⋮

User Instructions

SIZE: 018

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Load the program.			
2	Initialize.		[XEQ] WARI	SEED?
3	Key in a seed between 0 and 1 for the			
	random number generator.	seed	[R/S]	A 4,4,4,4,4,4
4	To see the other side of the board and/or			
	the score, press [R/S]. This can be done			
	as often as desired.		[R/S]	A (board)
				(or) B (board)
				(or) A= B=
5	Players A, B or the 41C moving B's counters			
	can move in any order with anyone starting			
	the game.			
	Player A	bin#	[A]	A (board)
	Player B	bin#	[B]	A (board)
	41C (moving for B)		[C]	A (board)
6	Go to step 4 or 5 as desired.			
7	To start a new game press [E] and go to			
	step 4 or 5 as desired.		[E]	A 4,4,4,4,4,4

Program Listings

```
 01◆LBL "WAR
I"                    Initialize
 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             41C move
 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            Offense
 31 RCL 16
 32 INT               41C pit #
 33 RCL IND           41C counters
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             Player pit #
 47 7
 48 X<=Y?
 49 GTO 00
 50 X<>Y
 51 R↑
 52 CHS
 53 RCL IND           Player counters
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            Found an
 81 INT               offensive move
 82 GTO A
 83◆LBL 00
 84 6
 85 STO 16
 86 0
 87 STO 15
 88◆LBL 04            Defense
 89 RCL 16            Player's pit #
 90 RCL IND           Player's
16                    counters
 91 X=0?
 92 GTO 00
 93 RCL X
 94 12
 95 /
 96 INT
 97 X>0?
```

Program Listings


```
 98 GTO 00
 99 RDN
100 +
101 12
102 MOD
103 X=0?
104 LASTX
105 7
106 X>Y?
107 GTO 00
108 RCL IND
Y
109 X=0?
110 GTO 00
111 3
112 X<=Y?
113 GTO 00
114 SF 07
115 R↑
116 RCL 15
117 X<=Y?
118 X<>Y
119 STO 15
120◆LBL 00
121 DSE 16
122 GTO 04
123 FC?C 07
124 GTO 00
125 RCL 15
126 GTO A
127◆LBL 00
128 6
129 XEQ 51
130 1
131 +
132 STO 15
133 STO 16
134◆LBL 03
135 6
136 +
137 RCL IND
X
138 X≠0?
139 GTO 00
140 RCL 15
141 1
142 -
143 6
144 MOD
145 X=0?
146 LASTX
```

41C pit #

41C counters

Defensive move found

Random move

```
147 RCL 16
148 X=Y?
149 GTO 01
150 RDN
151 STO 15
152 GTO 03
153◆LBL 01
154 SF 05
155 "NO MOVE
"
156 AVIEW
157 STOP
158 GTO 50
159◆LBL 00
160 RCL 15
161◆LBL B
162 6
163 +
164 CF 05
165◆LBL A
166 STO 15
167 STO 17
168 RCL IND
15
169 X=0?
170 GTO 50
171 STO 16
172 ST- IND
15
173◆LBL 08
174 RCL 17
175 1
176 +
177 12
178 MOD
179 X=0?
180 LASTX
181 STO 17
182 RCL 15
183 X=Y?
184 GTO 08
185 1
186 ST+ IND
17
187 DSE 16
188 GTO 08
189◆LBL 07
190 RCL 17
191 7
192 FS? 05
193 GTO 01
```

Player B

Player A

Distribute counters

Program Listings

```
194 X<=Y?
195 GTO 50
196 GTO 00
197◆LBL 01
198 X>Y?
199 GTO 50
200◆LBL 00
201 RCL IND
17
202 2
203 X>Y?
204 GTO 50
205 RDN
206 4
207 X<=Y?
208 GTO 50
209 RDN
210 FS? 05
211 ST+ 13
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
223 GTO 07
224◆LBL 50
225 SF 05
226 "A "
227 1.006
228 XEQ 00
229 PROMPT
230 "B "
231 7.012
232 XEQ 00
233 PROMPT
234 "A="
235 ARCL 13
236 "⊢,    B=
"
237 ARCL 14
238 PROMPT
239 GTO 50
240◆LBL 09
241 "⊢,"
```

Capture pits

Output

A board

B board

Score

```
242◆LBL 00
243 ARCL IND
 X
244 ISG X
245 GTO 09
246 RTN
247◆LBL 51
248 RCL 00
249 9821
250 *
251 .211327
252 +
253 FRC
254 STO 00
255 *
256 INT
257 RTN
258 .END.
```

Random number generator

70

80

90

00

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Seed	50	
	Bin A1		
	A2		
	A3		
	A4		
05	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	
40		90	
45		95	

STATUS

SIZE 018 TOT. REG. 82 USER MODE
ENG ____ FIX ____ SCI ____ ON ____ OFF ____
DEG ____ RAD ____ GRAD ____

FLAGS

#	INIT S/C	SET INDICATES	CLEAR INDICATES
05		A's move	B's move
06		Used	
07		Found one to attack	

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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:	Display:	
[XEQ] [ALPHA] SIZE [ALPHA] 004		
[///] [π] [STO] 00		
[XEQ] [ALPHA] SIMON [ALPHA]	HOW MANY?	
3 [R/S]	1	(sequence)
	NUMBERS?	
1 [R/S]	YES: 1	
	9	(sequence)
	2	
	NUMBERS?	
9 [R/S]	NO: 92, NOT 9	
	3	
	4	(sequence)
	6	
	NUMBERS?	
346 [R/S]	YES: 346	
	YOU MISSED 1	

User Instructions

SIZE:004

STEP	INSTRUCTIONS	INPUT	FUNCTION	DISPLAY
1	Key in program and store seed (0≤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"
02 FS? 00
03 RTN
04 RCL 00
05 9821
06 *
07 .211327
08 +
09 FRC
10 1.1111
11 *
12 FRC
13 STO 00
14 RTN
```
Random number generator

```
15◆LBL "SIM
ON"
16 FIX 0
17 CF 29
18 CF 06
19 FS?C 00
20 SF 07
21 XEQ "R"
22 FS?C 07
23 SF 00
24 "HOW MAN
Y?"
25 PROMPT
26 1 E3
27 /
28 1
29 +
30 STO 01
31 CLX
32 STO 02
```
Set display format

Clear error flag

Save status of F00

Ask for maximum length

Set up counters

```
33◆LBL 10
34 RCL 01
35 INT
36 XEQ "R"
37 XEQ "TON
ES"
38 XEQ "?"
39 FS? 05
40 GTO "NO"
41 "YES: "
42 ARCL X
43 AVIEW
44 ISG 01
45 GTO 10
46 FS?C 06
47 GTO 01
48 BEEP
```
Get a sequence

Display the sequence

Ask player what he saw

If wrong, branch to "NO"

Increment counter

Repeat loop

If any errors, skip win message

```
49 BEEP
50 "YOU WIN
"
51 AVIEW
52 RTN
```
Win message

```
53◆LBL "TON
ES"
54 RCL 01
55 INT
56 STO 03
57 RCL 00
58 FRC
59◆LBL 03
60 10
61 *
62 INT
63 VIEW X
64 TONE IND
X
65 LASTX
66 FRC
67 DSE 03
68 GTO 03
69 RTN
```
Set up tone counter

Begin tone loop

Decrement tone count

Repeat tone loop

```
70◆LBL "NO"
71 SF 06
72 TONE 2
73 TONE 0
74 "NO: "
75 ARCL X
76 "⊢, NOT
"
77 ARCL Y
78 AVIEW
79 PSE
80 1
81 ST+ 02
82 ISG 01
83 GTO 10
```
"Un-oh" sound

Increment error count

Increment counter

Repeat loop

```
84◆LBL 01
85 "YOU MIS
SED "
86 ARCL 02
87 CF 06
88 AVIEW
89 RTN
```
Error message

```
90◆LBL "?"
91 "NUMBERS
?"
92 PROMPT
93 CF 05
```
Ask player to input numbers seen

Program Listings

```
 94 RCL 00
 95 RCL 01
 96 INT
 97 10↑X
 98 *
 99 INT
100 X=Y?
101 RTN
102 SF 05
103 RTN
104 .END.
```

If correct, then done

Otherwise, set error flag

REGISTERS, STATUS, FLAGS, ASSIGNMENTS

DATA REGISTERS

00	Random sequence	50	
	length counter		
	error counter		
	tones counter		
05		55	
10		60	
15		65	
20		70	
25		75	
30		80	
35		85	
40		90	
45		95	

STATUS

SIZE 004 TOT. REG. 40 USER MODE

ENG ____ FIX 0 SCI ____ ON ____ OFF ____

DEG ____ RAD ____ GRAD ____

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

ASSIGNMENTS

FUNCTION	KEY	FUNCTION	KEY

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, minature 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 accomodate all programs.

GAMES

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

HEWLETT PACKARD

00041-90099 Rev. D 4/81 Printed in U.S.A