

Program Title THE GAME OF 3-D TIC-TAC-TOE

Contributor's Name ALAN STEIN

Address 372 SHOTWELL ST

City SAN FRANCISCO

State CA

Zip Code 94110

Program Description, Equations, Variables This program pits the HP 67 against a human opponent in the game of 3-D TicTacToe. The rules of this game are simple:

1) The board is a cube consisting 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 of her markers in a straight line. The line can lie in more than one level; e.g. diagonals are perfectly legitimate wins.

In short, the game is played just like regular TicTacToe, except that the board has one additional dimension, and is one square bigger in all dimensions. Unlike regular TicTacToe, 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 67 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} (R_{85}) are reserved for the game board. (see illustrations on next page).

Each square on the board can be characterized by its level=z, its row=y and its column=x. X,y,& z can have values from 1 through 4.

Operating Limits and Warnings When entering moves, make sure they are 3 digit decimal numbers. All three digits must be between 1 and 4. Entering a move outside this range may cause the program to make erroneous entries in the board, and blow up the game.

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

02925D Program Description I

Page 2 of 14

Program Title

3-d tictactoe

Contributor's Name

Address

City

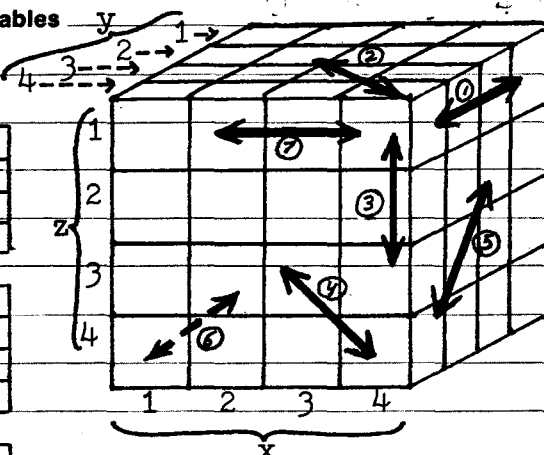
State

Zip Code

Program Description, Equations, Variables

LEVEL ROW REG.

(z)	(y)	#	BOARD
1	1	0	
1	2	1	
1	3	2	
1	4	3	
2	1	4	
2	2	5	
2	3	6	
2	4	7	
3	1	8	
3	2	9	
3	3	10	
3	4	11	
4	1	12	
4	2	13	
4	3	14	
4	4	15	



Note: The arrows with circled numbers indicate the seven kinds of win lines, and will be referred to later.

The table to the left shows how various combinations of level and row numbers correspond to various registers. Note that the value of x does not affect the register number.

Column(x) = 1 2 3 4

The player enters a move by providing the machine with the z,y & x positions for the move. The format is a:

3 digit decimal number : .ZYX.

DO NOT FORGET THE LEADING DECIMAL POINT when entering a move.

The machine will respond in the same form, with one exception-- if the machine wins, it will indicate this by outputting a move in the form 1.ZYX --the leading "1" indicates this is a winning move.

If the player makes a winning move, the machine will recognize this and halt during the execution of Card 1 with 4.000 in the display.

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Program Description I

Program Title _____

3-d tictactoe

Contributor's Name _____

Address _____

City _____

State _____

Zip Code _____

Program Description, Equations, Variables The 67 stores the moves in a row(register) of the board in the following form:

$N.0a0b0c0d$: N=total number of moves made so far

in that register(0 through 4). a,b,c,& d indicate the conditions of spaces (x-value) 4,3,2 and 1 in that row.

If a--d = 0, the space is empty. If a--d = 1, the space contains a machine's piece. If a--d = 5, the space contains a human player's piece.

For example: If $R_3 = 3.01000501$, this means that there are three pieces in the topmost front row(see illustration on preceding page). The first and last pieces are machine pieces, and the second space(x=2) has a player's piece in it. The third square is empty.

~~THE FOLLOWING SECTION DESCRIBES HOW THE COMPUTER DECIDES WHAT MOVE TO MAKE. Skip the rest of Program Description I, if you wish to play against the machine without knowing its strategy ahead of time.~~

~~The machine plays both a strategic game-- based on obtaining a strong overall position--and a tactical game, which either counters an immediate specific threat or takes immediate advantage of an unblocked line.~~

~~While both games are determined by the player's moves, the 67 does not actually look more than one move ahead.~~

~~There are two periods in a game. During the opening game, the machine will respond to a threatening situation produced by the human player, but will not attempt to to advantage of a strong tactical move that would advance its position. Rather, it makes strategic moves whenever possible, to lay out a good opening pattern of pieces.~~

~~During the main game, strategic moves are made only when there is no tactical move to be made.~~

~~The machine makes a tactical move whenever it finds that the last move (of either player) has produced an unblocked line of two or three pieces for a player. The 67 moves to either block an opponents line or~~

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Program Description I

Program Title 3-d tictactoe

Contributor's Name _____

Address _____

City _____

State _____

Zip Code _____

Program Description, Equations, Variables extend its own unblocked line. The 67 cannot remember from one move to the next. Hence, if it does not follow up on a tactical situation within one move, it will 'forget' about it, until another move causes a tactical search in that region. To keep running time down, the machine does not examine the entire board after each move. Instead it looks only at the win lines which are potentially changed by that move, (with minor exceptions which will be discussed later).

SEQUENCE OF OPERATIONS -- MAIN GAME

ENTRY OF PLAYER'S MOVE: The human player enters a 3-digit decimal fraction for her move, in the form .zyx: where z is the level of the move, y is the row, and x is the column. This number is stored in R_9 for future reference. The move is decomposed into its component values, which are stored as $R_B = y-1$, $R_C = z-1$, & $R_D = 4(z-1) + (y-1)$.

This is the number of the register containing that row of the board. $R_A = 10^{10-2x}$. This number corresponds to the position in the register of the desired move. $R_A * \text{reg}$ shifts the register to put the column corresponding to x in the first two integer digits to the left of the decimal. By taking INT, dividing by 100, and taking FRAC, we can then isolate those two digits. This procedure will be used frequently in the program.

Once the square is isolated, it is compared to 0. If it is $\neq 0$, then that move is illegal; i.e. the square is already filled, and the program attempts to jump to a nonexistent label, producing an error message. The player must then enter a correct move.

If the move is good, a 5 is entered in the appropriate location by adding $5/R_A$ to the reg. 1 is also added to the reg, to maintain the correct move count.

ANALYSIS OF HUMAN'S MOVE

There are seven types of possible win lines, as illustrated on pp2. They are:

- 1) A horizontal column, where x and z are constant
- 2) A horizontal diagonal, where z is constant. Note that there are two such diagonals, perpendicular to each other.

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Program Description I

Program Title _____			
3-d tictactoe			
Contributor's Name _____			
Address _____			
City _____	State _____	Zip Code _____	

Program Description, Equations, Variables 3) A vertical column; x and y are constant. 4) and 5) are two different kinds of vertical diagonals. To distinguish them, we will refer to (4) as a 'constant-y' diagonal and (5) as a 'constant-x' diagonal. Note that, like (2), there are two of each type, perpendicular to each other.

6) A full diagonal. There are four of these, running through the cube to opposite corners. Neither x, y, nor z is constant.

7) A row; y and z are constant.

For any given square on the board, (1), (3), and (7) exist. Also, at least one diagonal will exist. If the square lies on a full diagonal (6), then all diagonals are possible. Otherwise, only one diagonal, (2), (4), or (5) is possible.

It turns out that deciding which diagonals apply, while not difficult, takes longer than testing them all! The checking routines are designed to look at all types of diagonals. For those which have a constant term, it is set equal to the coordinate of the move; e.g. a move of .144 (in the illustration, the upper right near corner) would check (2) with z set = 1 (illustrated), (4) with y=4 (illustrated), and (5) with x=4 (illustrated). Note that the diagonals perpendicular to these would also be checked, even though they don't actually include the given move. Also, the four full-diagonals are checked every move.

The test of a line is performed by adding together the four appropriate squares. Since a player's move has a value of 5 and a 67 move, 1, there is a unique sum for each possible condition of the line. For example, a human's unblocked line must always have a sum divisible by 5. A machine's unblocked line must have a sum less than 5, etc.etc. In practice, the 67 is uninterested in all cases except unblocked lines of two or more pieces.

Only one subroutine is required to perform the summings and to test both the human and 67 win-lines. This subroutine is controlled by the following inputs:

R_1 holds the starting register in the line.

R_2 holds the amount by which R_1 must be incremented to

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Program Title

3-d tictactoe

Contributor's Name

Address

City

State

Zip Code

Program Description, Equations, Variables get to the next register in the line (e.g. for (3), $R_E = 4$; for (7), $R_E = 0$; for (5), $R_E = 3$ or 5)

F_0 indicates that a player move is being analyzed, and is used to control the comparison tests run on the final sum.

F_1 controls the summing process itself. If F_1 is off, then the registers are simply added together, meaning all the squares having the same x-value are summed. If F_1 is on, after each term is added, the sum is multiplied by 100, giving a final sum of the form

$10^6 * R(i) + 10^4 * R(i+e) + 100 * R(i+2e) + R(i+3e)$. The diagonal will be summed in the first two digits to the right of the decimal.

It turns out that lines (1) thru (6), comprising 12 distinct cases, can be defined by four sets of R_i and R_E . Three kinds of summing are done on each set: $F_1 = \text{off}$; $F_1 = \text{on}$, with the registers summed in ascending order; and $F_1 = \text{on}$, with the registers summed in descending order (which gives the two diagonals perpendicular to each other). Case (7) is treated as a separate case--note that the summing is done with F_1 on; i.e. it is treated as a diagonal.

Once the sum is completed, and the desired two-digit sum isolated, it is divided by 5 and a remainder looked at. If it's 0, then the row is unblocked. The quotient gives the number of pieces in the row. This number is compared with R_{s7} , which holds the highest such number found in the round (and is initialized to 2, at the start of the round). If the quotient is $\geq R_{s7}$, it replaces the old R_{s7} . In addition, F_2 is set, to indicate a tactical move was found, and the # of this particular test is recalled from R_{s8} and stored in R_{s6} .

The 67 repeats this sum and check procedure until all win-lines have been examined.

ANALYSIS OF MACHINE'S MOVE

The testing procedure is identical with that of the previous section. F_0 is cleared to indicate different test criteria for eliminating blocked lines (and to skip the /-by-5 section). F_2 is cleared: since it will be reset if a machine tactical status replaces a player's status it provides an indicator for later that we are dealing with a machine offense rather than a defense. F_3 is cleared to prepare for later use.

This program has been verified only with respect to the numerical example given in Program Description II. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

02925D Program Description I

Page 7 of 14

Program Title _____

3-d tictactoe

Contributor's Name _____

Address _____

City _____

State _____

Zip Code _____

Program Description, Equations, Variables The previous 67 move is recalled and parsed, so that the correct x,y, and z values are available.

The test subroutine LBL 0 is run.

PREPARE FOR CARD 2

F₂ is checked--if it is off, then F₃ is set and the player's move is parsed to load the x,y, and z values into the registers.

The test # in R₆ is recalled and analyzed to tell the 67 which line was the critical one. R₁ is loaded, and F₀ & F₁ are set appropriately for the routines on Card 2.

Execution then jumps to LBL C, which is a loop containing a MERGE/PAUSE sequence. The MERGE retains the flag statuses set by part 1. LBL C loops until Card 2 is read in.

DECIDE ON MACHINE'S NEW MOVE

Card 2 begins with a GTO(i) which jumps to any of 6 routines. 1 thru 5 simply initialize the 67 to check for open spaces in a win-line defined by the test #. It is similar to the procedure used to sum a specific win-line. This will be discussed in more detail, later.

A "0" (GTO LBL 0) indicates that no tactical position passed the test criteria. In this case, a strategic move is made. The 67 selects the move in the following fashion :

Each of the levels of the board is summed, by adding the registers together, starting with level 2. The level having the least # of moves on it is selected. If two levels have the same number, the first one found is the chosen one. This favors the inner levels.

The same sort of selection process is performed on the files (the column of registers having y constant), starting with file 2. The intersection of the chosen level and file defines one register in which the machine will attempt to move. Depending on the reg.#, one of two sequences of square-searching is selected; this provides a better layout of pieces, then would be gotten from using the same pattern all the time.

Each square in the register is checked to see if it is empty, in a manner similar to that used to check the validity of the player's move.

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Program Description I

Program Title

3-d tictactoe

Contributor's Name

Address

City

State

Zip Code

Program Description, Equations, Variables For a tactical move, registers and squares within a register are examined along the chosen win-line. The selection procedure for the registers is identical with that in the original testing routine. Instead of summing the registers, though, R_A is used to isolate the appropriate square, and it is tested to see if it is empty, in the fashion previously described. If a diagonal is being examined ($F_1=on$), no staggered sum is performed, rather, R_A is multiplied or divided by 100, as is appropriate, between registers.

RECORDING THE MACHINE'S MOVE

Once a satisfactory move is found, $1/R_A$ is added to the appropriate register, loading a '1' into the appropriate square. 1 is added to the register to update the move counter. Then, x is reconstructed from R_A and the coordinates combined to produce $.zyx$. This is stored in R_{s8} . The tactical status in R_7 is recalled, divided by 3 and the INT taken. If $R_7=3$, this produces a value of 1, otherwise, it =0. F_2 is checked. If it is off, it means that this was an offensive move and maybe the machine has moved in a row which had 3-in-a-row. That is, the 67 has won. If this is the case, then the value added to $.zyx$ will be 1 and the output will be of the form $1.zyx$. If this is not the case 0 will be added, and the output will be the standard one.

OPENING GAME

The opening game differs in the emphasis given tactics, and specifically in the use of R_{s8} . If $R_{s8}=0$, it is the first round, and the 67

skips the tactical analysis of Card 1. R_{s8} is used as a round counter = $.01 \times \#$ of rounds. So long as the count is less than .06, F_2 is set before Card 2 is loaded. F_2 inhibits the writing of the machine move in R_{s8} . During the opening game, the machine also skips the tactical analysis of its own moves, to emphasize strategy, but it does examine the player's move, starting in round 2, to look for threats. So long as $R_{s8} < .1$, the 67 plays opening game. Once a machine move is written into R_{s8} the register, the machine will skip the opening game variant.

This completes the description of the machine's playing pattern.

This program has been verified only with respect to the numerical example given in *Program Description II*. User accepts and uses this program material AT HIS OWN RISK, in reliance solely upon his own inspection of the program material and without reliance upon any representation or description concerning the program material.

NEITHER HP NOR THE CONTRIBUTOR MAKES ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH REGARD TO THIS PROGRAM MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER HP NOR THE CONTRIBUTOR SHALL BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE FURNISHING, USE OR PERFORMANCE OF THIS PROGRAM MATERIAL.

Diagram illustrating the layout of the playing board for different values of z (1, 2, 3, 4). The board is a 4x4 grid. The columns are labeled 1, 2, 3, 4, and the rows are labeled $y=1$, $y=2$, $y=3$, $y=4$. A bracket under the columns is labeled x . The boards are arranged horizontally, with the label "PLAYING BOARD" centered below them.

Sample Problem(s) _____ A SAMPLE GAME IS PLAYED OUT HERE. This is a real game; since the program is deterministic, if you make the same moves, the machine will make the same responses as these.

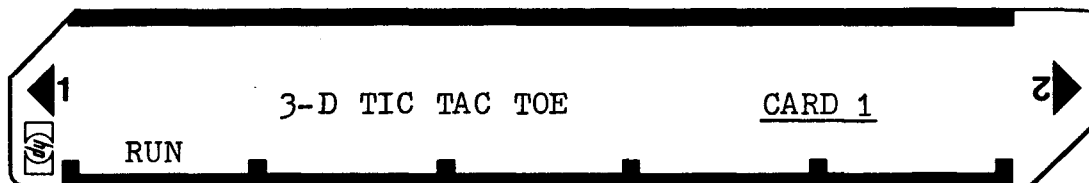
<u>move</u>	<u>ALAN</u>	<u>67</u>	<u>COMMENTS</u>
1	.231	.322	Unusual human opening.
2	.414	.142	both sides are trying to set up position
3	.243	.323	
4	.324	.144	machine blocks two-in-row
5	.212	.111	machine blocks two-in-row
6	.133	.434	human blocks two-in-row
7	.332	.422	
8	.214	.233	human tries to set up trap, machine does so 1st.
9	.411	.143	aaarrrrggghhh
10	.413	1.141	MACHINE WINS!

In general, the machine is more likely to win a short game, the human, a long one.

Solution(s)

Figure 1 shows four 3x3 grids illustrating different configurations of 'X' and 'O' marks. The grids are labeled 1, 2, 3, and 4. Grid 1 shows 'X' marks at (1,1), (3,1), (3,2), (3,3), (3,4), and (3,5). Grid 2 shows 'O' marks at (1,2), (1,4), (2,2), and (3,2), and an 'X' mark at (2,3). Grid 3 shows 'X' marks at (2,2) and (2,3), and 'O' marks at (2,4) and (3,2). Grid 4 shows 'O' marks at (1,2), (1,4), and (1,5), and 'X' marks at (2,3) and (3,4).

Reference(s)



CARD 2 SAYS ONLY "3-D TIC TAC TOE : CARD 2". (NO USER CONTROLLED FUNCTIONS)

[illegible]

STEP	KEY ENTRY	KEY CODE	COMMENTS	STEP	KEY ENTRY	KEY CODE	COMMENTS
	-	51	z-1		+	61	add registers.
	4	04		170	x ↔ I	35 24	
	x	71	$R_c = 4(z-1)$		RCL E	34 15	increment I
	STO C	33 13	Reg #		+	61	
	+	61	$4(z-1)+(y-1)$		x ↔ I	35 24	continue stag-
	STO D	33 14	$R_D = \text{Reg \#}$		F ? 1	35 71 01	gered summing
	RTN	35 22	INITIALIZE TEST		x	71	for diagonals
120	* LBL 0	31 25 00	CONTROLS		RCL (i)	34 24	end sum is
	1	01	There are 5		+	61	reg1+reg2+reg3+
	RCL C	34 13	classes of reg-		x ↔ I	35 24	reg4 or 106reg1
	GSB 7	31 22 07	isters correspon-	180	RCL E	34 15	+104reg2+102reg3
	4	04	ding to possible		+	61	+reg4. (i.e staggered
	RCL B	34 12	win-lines. All		x ↔ I	35 24	sum).
	GSB 7	31 22 07	but the last are		F ? 1	35 71 01	Note that "single
	5	05	associated with		x	71	row" test "staggered"
	ENTER	41	3 possible ori-		RCL (i)	34 24	Normalize non-
	0	00	entations of		+	61	diagonal_sum=S
130	GSB 7	31 22 07	such lines. This		F ? 1	35 71 01	100/A=10 ^{x-8}
	3	03	routine sets up		GTO D	22 14	
	ENTER	41	the classes		R↑	35 54	
	GSB 7	31 22 07	for LBL 7 to	190	RCL A	34 11	
	0	00	look at.		/	81	
	STO E	33 15	This test looks		/	81	Snew=Sold*10 ^{8-x}
	RCL ED	34 14	only for win in		* LBL D	31 25 14	ANALYZE SUM --
	ST I	35 33	a single row.		FRAC	32 83	Isolate 1st two
	GTO 9	22 09	SET TEST POINTS		R↑	35 54	decimal digits-
	*LBL 7	31 25 07	non-diagonal test		x	71	which are sum of
140	CF 1	35 61 01	1st register in		INT	31 83	moves in tested line
	ST I	35 33	row.		P ↔ S	31 42	set 2nd reg's.
	R↓	35 53	Interval between		4	04	y=4, x=Sum
	STO E	33 15	TEST!		x ↔ y	35 52	If looking at
	GSB 9	31 22 09	Set diagonals	200	F ? 0	35 71 00	player, then skip
	SF 1	35 51 01	reverse interval		GTO 2	22 02	to LBL 2.
	RCL E	34 15	to count back to		x > y?	32 81	If sum > 4, then line
	CHS	42	start		RTN	35 22	is blocked by player.
	STO E	33 15	TEST!		GTO D	22 14	
	GSB 9	31 22 09	Count up; check		* LBL 2	31 25 02	ANALYZE PLAYER
150	RCL E	34 15	diag running		5	05	SUM
	CHS	42	other way.		/	81	y=4, x=Sum/5
	STO E	33 15	TESTING CYCLE		x = y?	32 51	If true, then play-
	*LBL 9	31 25 09	Increment test#		R/S	84	er has WON.
	1	01	set primary reg's.	210	FRAC	32 83	If not integer,
	STO+8	33 61 08	fill stack with		x ≠ 0?	31 61	then line is
	P ↔ S	31 42	100's		RTN	35 22	blocked by mach.
	EEX	43			LST x	35 82	recall Sum/5
	2	02			*LBL D	31 25 14	TEST TAC STATUS
	ENTER	41			RCL 7	34 07	If old status
160	ENTER	41	pull first reg.		x > y?	32 81	is higher than
	RCL (i)	34 24	increment I		RTN	35 22	Sum, end test.
	x ↔ I	35 24			x ↔ y	35 52	If not, SF 2, to
	RCL E	34 15			SF 2	35 51 02	indicate tact.
	+	61		220	STO 7	33 07	Store Sum as new
	x ↔ I	35 24			RCL 8	34 08	tactical status.
	F ? 1	35 71 01	if looking at		STO 6	33 06	store test # in
	x	71	diagonals, x 100.		RTN	35 22	R6.
	RCL (i)	34 24	next register.		---	84	

LABELS					FLAGS	SET STATUS		
A RUN	B parse move	C Merge #2	D used	E -----	Player move	FLAGS	TRIG	DISP
a -----	b -----	initial. G 2	d -----	e -----	1 diagonals	0 <input type="checkbox"/> ON <input checked="" type="checkbox"/> OFF	DEG <input checked="" type="checkbox"/>	FIX <input checked="" type="checkbox"/>
Set tests	1 -----	2 used	3 -----	4 -----	2 used	1 <input type="checkbox"/> <input checked="" type="checkbox"/>	GRAD <input type="checkbox"/>	SCI <input type="checkbox"/>
5 -----	6 -----	Set tests	8 -----	9 run test	mach. move	2 <input type="checkbox"/> <input checked="" type="checkbox"/>	RAD <input type="checkbox"/>	ENG <input type="checkbox"/>
						3 <input type="checkbox"/> <input checked="" type="checkbox"/>		n. 3

STEP	KEY ENTRY	KEY CODE	COMMENTS	STEP	KEY ENTRY	KEY CODE	COMMENTS
	RCL E	34 15	Increment index		INT	31 83	chosen space is
	+	61		170	EEX	43	in integer range
	x ↔ I	35 24			2	02	Take int. part,
	RCL(i)	34 24	add next register		/	81	divide by 100
	+	61			FRAC	32 83	take fraction,
	x ↔ I	35 24	When done, all		x ≠ 0?	31 61	to isolate. If
	RCL E	34 15	four registers		RTN	35 22	≠ 0, then move is
120	+	61	representing a given		RCL A	34 11	no good. Other-
	x ↔ I	35 24	level or file		1/x	35 62	wise, store 1 in
	RCL(i)	34 24	of the board will		STO+(i)	33 61 24	that position
	+	61	be summed. The		ISZ(i)	32 34	for machine's
	x ↔ I	35 24	integer part of	180	log	31 53	move & add 1 to
	RCL E	34 15	the sum is the		2	02	integer part, to
	+	61	total # of moves		/	81	record total
	x ↔ I	35 24	made so far, on		5	05	moves.
	RCL(i)	34 24	that level or file		+	61	x=log(A)/2 +5
	+	61			RC I	35 34	I=4(z-1)+y-1
130	x ↔ I	35 24			4	04	
	RCL E	34 15			/	81	integer part=
	+	61			INT	31 83	z-1; frac part =
	x ↔ I	35 24			IST x	35 82	y-1/4
	INT	31 83	# of moves.	190	FRAC	32 83	
	RCL A	34 11	previous low #.		4	04	
	x ≤ y ?	32 71	if new # is no		x	71	y-1
	RTN	35 22	lower, then return		1	01	
	R ↓	35 53	otherwise, store		+	61	y
	STO A	33 11	new low #		x ↔ y	35 52	z-1
140	R ↓	35 53	and store appro-		1	01	
	STO C	33 13	prpriate y-1 or z-1.		+	61	z
	RTN	35 22			EEX	43	
	* LBL 6	31 25 06	FIND EMPTY SPACE		1	01	10
	EEX	43		200	x	71	10z
	2	02	Look for empty		+	61	10z+y
	GSB c	32 22 13	spaces in the		EEX	43	
	EEX	43	order: 4th, 3rd,		1	01	10
	4	04	1st, and finally		x	71	100z+10y
	GSB c	32 22 13	2nd space in		+	61	100z+10y+x
150	EEX	43	row chosen by		EEX	43	
	8	08	LBL 0 and E.		3	03	1000
	GSB c	32 22 13			/	81	output = .zyx
	* LBL 7	31 25 07	FIND EMPTY SPACE		F ? 2	35 71 02	if F ₂ is on, it
	EEX	43	Look in the	210	R/S	84	is still open-
	6	06	order: 2nd, 3rd,		P ↔ S	31 42	ing game; R ₈ is
	GSB c	32 22 13	4th, and 1st.		STO 8	33 08	being used to
	EEX	43			RCL 7	34 07	store # of round.
	4	04			P ↔ S	31 42	Otherwise, store
	GSB c	32 22 13			3	03	mach. move in R ₈
160	EEX	43			/	81	
	2	02			INT	31 83	recall old tact
	GSB c	32 22 13			F ? 3	35 71 03	status. If = 3,
	EEX	43			CLX	44	then if F ₃ =on,
	8	08		220	+	61	indicating
	* LBL c	32 25 13	CHECK/STORE MOVE		R/S	84	machine tact.
	STO A	33 11	recall chosen reg		↓	↓	status, this is
	RCL(i)	34 24	and shift so that		↓	↓	a machine win.
	x	71			↓	↓	Add 1 to output

LABELS				FLAGS		SET STATUS		
A -----	B sum rows	C -----	D test tests	E -----	choose diagonal	FLAGS	TRIG	DISP
find move	b -----	test move	d -----	sum rows	1 set diagonal	ON OFF	DEG <input checked="" type="checkbox"/>	FIX <input checked="" type="checkbox"/>
strategy	1 tactic	2 tactic	3 tactic	4 tactic	2 openings	0 <input type="checkbox"/> <input checked="" type="checkbox"/>	GRAD <input type="checkbox"/>	SCI <input type="checkbox"/>
tactic	6 try move	7 try move	8 -----	keep i<16	3 mach. move	1 <input type="checkbox"/> <input checked="" type="checkbox"/>	RAD <input type="checkbox"/>	ENG <input type="checkbox"/>
						2 <input type="checkbox"/> <input checked="" type="checkbox"/>		n <u>3</u>
						3 <input type="checkbox"/> <input checked="" type="checkbox"/>		