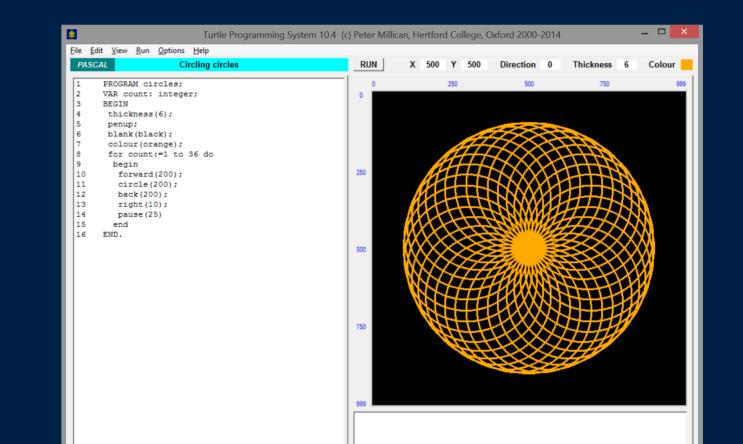
The Turtle System www.turtle.ox.ac.uk

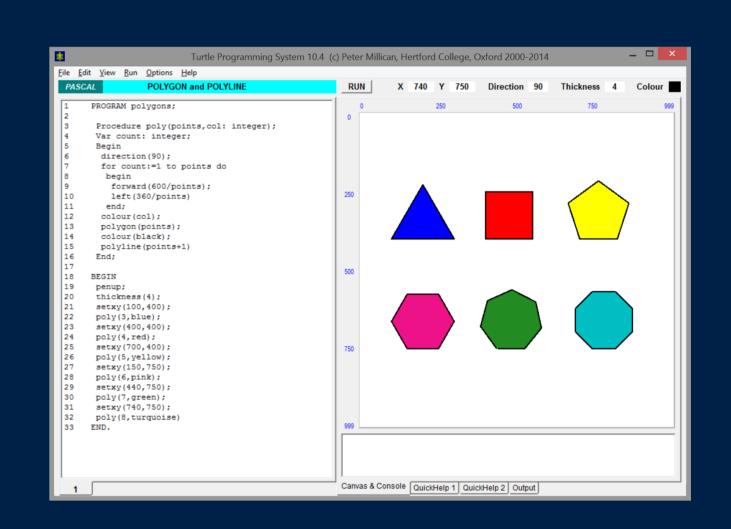
An easy tool for teaching BASIC, Java, Pascal, and Python, with selflearning exercises and study materials, developed at Oxford University with matched funding from the Department for Education

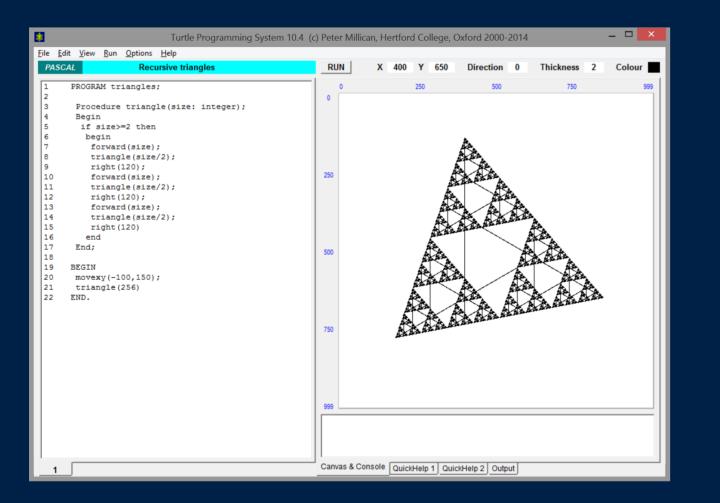
- Free, safe, and well-supported software
- Suitable for complete beginners both teachers and students
- Programming is intuitive and visual

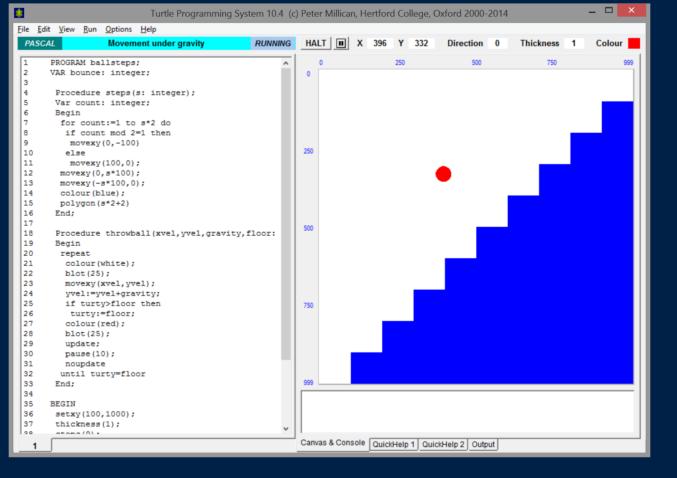


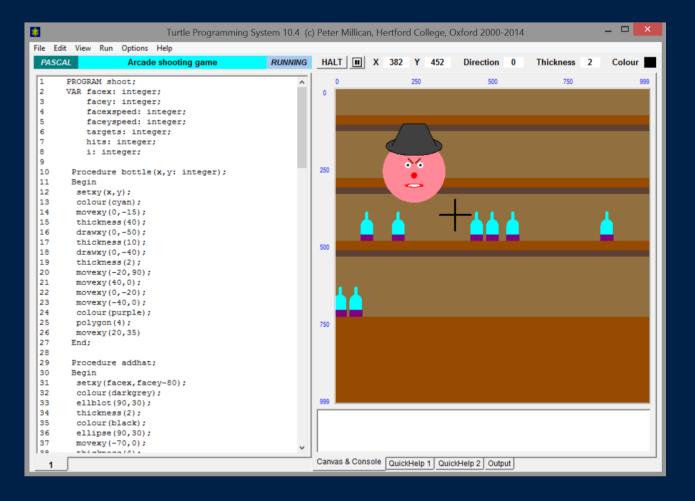
Canvas & Console QuickHelp 1 QuickHelp 2 Outpu

- Meets National Curriculum requirements
- Create animations, interactive games, and even artificial intelligence programs
- Apps can run on webpages and mobiles
- A single downloadable file, no installation (and no hidden effect on your machine)









File Edit View Analysis Compile Run Options Language Help

4 VAR board: array[011] of integer; numgames: integer; 5 numgames: integer; 6 randfirst: boolean; 7 i,column,row: integer; 9 Procedure doblot(x,y,col: integer); 10 Begin 11 setxy(x*250+250,y*250+175); 12 colour(col); 13 blot(70) 14 End; 15 Begin 16 Procedure copy(var fromarray.toarray: array of roarray[1]:=fromarray[1] 16 For i:=0 to fromarray.length-1 do toarray[1]:=fromarray[1] 21 End; 23 Procedure evaluate(var bl: array of integer; replyscore: integer; 24 Var mover: integer; 25 replyscore: integer; 26 b2; array[01] of integer; 27 minteger 28 Begin 29 mover::cross+nought-jm; 20 0 0 0 0 0 0 40 1 29 mover::cross+nought-jm; integer global 62to 119 47 to 55 27 mainteger; integer glo	PASC	AL Noughts and crosses RUNNI	NG	HALT	ъx	250	Y 17	5 D)irectior	n 0	Thi	ckness	10	Colour
3 nought=2; 4 VAR board: array[0.11] of integer; 5 numgames: integer; 6 randfirst: boolean; 1, column,row: integer; 8 9 Procedure doblot(x,y,col: integer); 10 setsy(x*250+250,y*250+175); 12 0 13 blot(70) 14 End; 15 0 16 Procedure copy(var fromarray,toarray: array of integer;); 18 Begin 19 for i=0 to fromarray.length-1 do 20 lowed[0], boxed[0], boxed[10, boxed[11], mumgame: models: i column rest; 18 Begin 19 for i=0 to fromarray.length-1 do 20 o 0 0 21 30 2 -1 36 22 Procedure evaluate(var b1: array of integer; 1 -1 -1 23 Procedure evaluate(var b1: array of integer; 1 -1 -1 24 Var mover: integer; 100 -1 -1 -1 25 replyscore: intoteger; -1 -1 </td <td></td> <td></td> <td>^</td> <td>Display:</td> <td>🔽 Me</td> <td>emory Sta</td> <td>ick 🗆</td> <td>Memory</td> <td>Heap</td> <td>🔽 Vari</td> <td>ables Ta</td> <td>able</td> <td>Show Cu</td> <td>rrent Stat</td>			^	Display:	🔽 Me	emory Sta	ick 🗆	Memory	Heap	🔽 Vari	ables Ta	able	Show Cu	rrent Stat
4 VAR board: array[01] of integer; numgames: integer; f randfirst: boolean; i,column,row: integer; 0 Aude Newful Newful Newful Newful Newful Laws the stress of th				MStock	0	4	2	2	4	5	6	-	0	9
S numgames: integer; 6 randfirst: boolean; 7 i,column,row: integer; 8					0	-	_	_		_	0	1	8	9
6 randfirst: boolean; 7 i,column,row: integer; 9 Procedure doblot(x,y,col: integer); 10 Begin 11 setxy(x*250+250,y*250+175); 12 colour(col); 13 blot(70) 14 End; 15 Procedure copy(var fromarray.toarray: array of integer;) 18 Begin 19 for i:=0 to fromarray.length-1 do 10 tarray of integer; 12 0 <t< td=""><td></td><td></td><td></td><td>00</td><td></td><td></td><td></td><td>^subr 2</td><td></td><td></td><td></td><td></td><td></td><td>turt</td></t<>				00				^subr 2						turt
7 i.column.row: integer; 8 Procedure doblot(x,y,col: integer); 10 Begin 11 setxy(x*250+250,y*250+175); 12 colour(col); 13 blot(70) 14 End; 15 Frocedure copy(var fromarray.toarray: array of to tarray[i]:=fromarray[i] 16 Procedure evaluate(var b1: array of integer;) 17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 12 toarray[i]:=fromarray[i] 21 End; 22 Procedure evaluate(var b1: array of integer;) 24 Var nover:integer; 25 replyscore:integer; 26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 29 mover:=cross+nought-jm; 29 mover:=cross+nought-jm; 29 b1[1]:==dp:th; 31 i((b2[0]=m) and (b1[1]=jm) and (b1[2]=jm) 33 b1[1]:=dp:f global 6210 119 34 if ((lb1[0]=m)					5	800	0	0	129	5	250	175	0	10
Procedure doblot(x,y,col: integer); Begin 1 setxy(x*250+250,y*250+175); colour(col); blot(70) 4 End; 5 Procedure copy(var fromarray,toarray: array of 1 blot(70) 4 End; 5 Procedure copy(var fromarray,toarray: array of 1 var i: integer; 8 Begin 9 for i:=0 to fromarray.length-1 do 1 varray[i]:=fromarray[i] 2 Procedure evaluate(var b1: array of integer; 7 wr integer; 8 Begin 9 rocedure evaluate(var b1: array of integer; 7 m: integer; 8 Begin 9 rocedure evaluate(var b1: array of integer; 7 m: integer; 8 Begin 9 rocedure evaluate(var b1: array of integer; 7 m: integer; 9 b2: array[011] of integer; 9 mover:=cross+nought-jm; 9 replyscore:=-2; 9 mover:=cross+nought-jm; 9 for (1b10]:=-1; 9 h1[1]:=depth; 9 mover:=cross+nought-jm; 9 nd (b1[7]=jm) and (b1[1]=jm) and (b1[2]=jm 9 nd (b1[7]=jm) and (b1[1]=jm) and (b1[2]=jm 9 nd (b1[7]=jm) and (b1[1]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[1]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=m) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=jm) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=m) and (b1[6]= 7 nd (b1[7]=jm) and (b1[6]=jm) and (b1[6]= 7 nd (b1[7]=jm) and														
Image: control of contro	8			10	turtc	board		board[0]	board[1]	board[2]	board[3]	board[4]	board[5]	board[6]
10 Begin 11 setxy(x*250+250,y*250+175); 12 colour(col); 13 blot(70) 14 End; 15 Procedure copy(var fromarray.toarray: array of to array[1]:=fromar	9	<pre>Procedure doblot(x,y,col: integer);</pre>			33	12	12	33	2	0	0	0	0	0
12 colour(col); 13 bloc(70) 14 End; 15 16 Procedure copy(var fromarray, toarray: array of 17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 20 toarray[i]:=fromarray[i] 21 End; 23 Procedure evaluate(var b1: array of integer; 24 Var mover: integer; 25 replyscore: integer; 26 b2: array[011] of integer; 77 m: integer; 28 Begin 29 mover::=cross+nought-jm; 26 b2: array[011] of integer; 77 m: integer; 28 Begin 29 mover:=cross+nought-jm; 20 nover:=cross+nought-jm; 21 integer; global 62 to 119 29 mover:=cross+nought-jm; 20 nover:=cross+nought-jm; 21 if ((1010]==jm) and (b1[3]==jm) 23 b1[10]:=-1; 33 b1[11]:=depth; 34	10				1.0			1.0						
12 colour(col); 13 blot(70) 14 End; 15 Procedure copy(var fromarray,toarray: array of 16 Procedure copy(var fromarray,toarray: array of 17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 20 toarray[i]:=fromarray[i] 21 End; 23 Procedure evaluate(var bl: array of integer;) 24 Var mover: integer; 25 replyscore: integer; 26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover::cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[3]=jm)) or ((b1[0])=jm) and (b1[6]=jm) and	11	setxy(x*250+250,y*250+175);		20	board[7]	board[8]	board[9]	board[10]	board[11]	numoame	randfirst	i	column	row
14 End; 15 Procedure copy(var fromarray,toarray: array of 16 Procedure copy(var fromarray,toarray: array of 17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 20 0 0 0 0 21 End; 23 Procedure evaluate(var b1: array of integer;) integer global 62 to 119 47 to 95 24 Var mover: integer; integer; global 62 to 119 47 to 95 25 replyscore: integer; integer; global 62 to 119 47 to 95 26 b2: array[011] of integer; integer; global 62 to 119 47 to 95 26 b2: array[011] of integer; integer; global 62 to 119 47 to 95 27 m: integer; integer; global 62 to 119 47 to 95 28 Begin - integer global 62 to 119 47 to 95 29 mover:=cross+nought-jm; - integer global 62 to 119 47 to 95 31 b1[9	12	colour(col);						2	4	1				0
15 30 12 33 0 2 -1 36 12 33 2 16 Procedure copy(var fromarray, toarray: array of integer;) 12 33 0 2 -1 36 12 33 2 17 Var i: integer;) 18 Begin 9 for i:=0 to fromarray.length-1 do 2 0 0 0 0 0 40 2 0 0 0 0 4 7 20 toarray[1]:=fromarray[1] 1 End; 1 <td>13</td> <td>blot(70)</td> <td></td> <td></td> <td>-</td> <td>-</td> <td></td> <td>-</td> <td></td> <td></td> <td>-</td> <td></td> <td>-</td> <td></td>	13	blot(70)			-	-		-			-		-	
15 16 Procedure copy(var fromarray,toarray: array of Var i: integer; 16 Begin 19 for i:=0 to fromarray.length-1 do 20 toarray[1]:=fromarray[1] 21 End; 22 0 0 0 0 0 4 21 End; 10 10 <	14	End;		20										
16 Procedure copy (var fromarray, toarray: array of var i: integer; 17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 20 toarray[i]:=fromarray[i] 21 End; 22 0 0 0 0 0 0 0 23 Procedure evaluate (var b1: array of integer; iurtx (0) PROGRAM 6 - integer global 62 to 119 47 to 95 24 Var mover: integer; 10 - integer global 62 to 119 47 to 95 25 replyscore: integer; 10 - integer global 62 to 119 47 to 95 26 b2: array[011] of integer; 10 - integer global 62 to 119 47 to 95 27 m: integer; global 62 to 119 47 to 95 10 10 - integer global 62 to 119 47 to 95 28 Begin - mover:=cross+nought-jm; - integer global 62 to 119 47 to 95 30	15			- 30	12	22	0	2	1	26	12	22	2	0
17 Var i: integer; 18 Begin 19 for i:=0 to fromarray.length-1 do 20 toarray[i]:=fromarray[i] 21 End; 22 Procedure evaluate(var b1: array of integer;) 23 Procedure evaluate(var b1: array of integer;) 24 Var mover: integer; 25 replyscore: integer; 26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[10]:=-1; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[2]=jm) and (b1[2]=jm) and (b1[6]=m)) or ((b1[2]=jm) and (b1[6]=m)) or ((b1[6]=m)) and (b1[6]=m))		Procedure copy(var fromarray,toarray: array c			12	- 35	U	2	-1		12	35	2	0
10 for i=0 to fromarray.length-1 do 19 for i=0 to fromarray[i] 21 End; 23 Procedure evaluate (var b1: array of integer;) 24 Var mover: integer; 25 replyscore: integer; 26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[5]=m)) or ((b1[2]=jm) and (b1[6]=m) and (b1[6]=m) and (b1[6]=m) and (b1[6]=m) and (b1[6]=m) and (b1[6]=m)		Var i: integer;				:				Ŷ				_
101 1		Begin		40									-	
21End;22Procedure evaluate(var b1: array of integer;)23Procedure evaluate(var b1: array of integer;)24Var mover: integer;25replyscore: integer;26b2: array[011] of integer;27m: integer;28Begin29mover:=cross+nought-jm;30replyscore:=-2;31b1[9]:=0;32b1[10]:=-1;33b1[11]:=depth;34if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm)36and (b1[7]=jm) and (b1[4]=jm) and (b1[6]=37or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=					2	0	0	0	0	0	0	4	1	3
22 100 10														
23 Procedure evaluate(var b1: array of integer;) 24 Var mover: integer; 25 replyscore: integer; 26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm)) 36 and (b1[7]=jm) and (b1[4]=jm) and (b1[6]= 37 or (b1[2]=jm) and (b1[4]=jm) and (b1[6]= 37 or (b1[2]=jm) and (b1[4]=jm) and (b1[6]=		End;		Identifier				Array	Туре	Refe	rence L	ine Scop		
24 Var mover: integer; 1					(0) P	ROGRAM		-	intege	r glo				
25 replyscore: integer; integer (0) PROGRAM 9 - integer (0) pland 62 to 119 47 to 95 26 b2: array[011] of integer; integer; global 62 to 119 47 to 95 27 m: integer; global 62 to 119 47 to 95 28 Begin mover:=cross+nought-jm; integer global 62 to 119 47 to 95 30 replyscore:=-2; integer global 62 to 119 47 to 95 31 b1[9]:=0; integer global 62 to 119 47 to 95 32 b1[10]:=-1; stillop:=n); if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[2]=jm)) or ((b1[2]=jm) and (b1[6]=m)) or ((b1[i					-	intege					
26 b2: array[011] of integer; 27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm) 36 and (b1[7]=jm) and (b1[4]=jm) and (b1[6]= 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=								-	intege	r glo				
27 m: integer; 28 Begin 29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 bl[9]:=0; 32 bl[10]:=-1; 33 bl[11]:=depth; 34 if (((bl[0]=jm) and (bl[1]=jm) and (bl[2]=jm) 35 and (bl[7]=jm) and (bl[2]=jm) and (bl[6]= 36 and (bl[7]=jm) and (bl[4]=jm) and (bl[6] 37 or ((bl[2]=jm) and (bl[4]=jm) and (bl[6]=					x - y -			-						
28 Begin numgames (0) PROGRAM 25 - integer global 62 to 119 47 to 95 29 mover:=cross+nought-jm; 30 replyscore:=-2; iii (0) PROGRAM 26 - boolean global 62 to 119 47 to 95 31 b1[9]:=0; column (0) PROGRAM 27 - integer global 62 to 119 47 to 95 32 b1[10]:=-1; column (0) PROGRAM 28 - integer global 62 to 119 47 to 95 33 b1[11]:=depth; (1) (0) PROGRAM 28 - integer global 62 to 119 47 to 95 34 if (((b10]=jm) and (b1[1]=jm) and (b1[2]=jm) column (0) PROGRAM 28 - integer global 62 to 119 47 to 95 35 and (b1[7]=jm) and (b1[8]=jm) or ((b1[0]) row (0) PROGRAM 29 - integer global 62 to 119 47 to 95 36 and (b1[7]=jm) and (b1[8]=jm) or ((b1[0]) row (0) PROGRAM 2 - integer value 10 to 14 5 to 11								-		r glo	bal 6	2 to 119		
29 mover:=cross+nought-jm; 30 replyscore:=-2; 31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm) 35 and (b1[7]=jm) and (b1[1]=jm) and (b1[2]=jm 36 and (b1[7]=jm) and (b1[4]=jm) and (b1[6]= 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=					x - y -			[011]	-					
30 replyscore:=-2; iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii				-				-	-					
31 b1[9]:=0; 32 b1[10]:=-1; 33 b1[11]:=depth; 34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm) 35 and (b1[7]=jm) and (b1[8]=jm)) or ((b1[2]=jm) and (b1[5] 36 and (b1[7]=jm) and (b1[4]=jm) and (b1[6]= 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=				randfirst						3.4				
32 b1[10]:=-1; form b1[10]:=-1; form b1[11]:=depth; 33 b1[11]:=depth; form (0) PROGRAM 29 - integer global 62to 119 47 to 95 34 if ((b10]=jm) and (b1[1]=jm) and (b1[2]=jm (1) doblot 1 - integer value 10 to 14 5 to 11 35 and (b1[7]=jm) and (b1[3]=jm)) or ((b1[0]) (1) doblot 2 - integer value 10 to 14 5 to 11 36 and (b1[7]=jm) or ((b1[2]=jm) and (b1[6]) or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]) 1 - integer value 10 to 14 5 to 11 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]) toarray (2) copy 2 [02] integer refarray 18 to 21 12 to 20				i						-				
33 bl[11]:=depth; 34 if (((bl[0]=jm) and (bl[1]=jm) and (bl[2]=jm 35 and (bl[7]=jm) and (bl[8]=jm)) or ((bl[0] 36 and (bl[7]=jm) or ((bl[2]=jm and (bl[6]=m) and (bl[6]=m)) or ((bl[6]=m) and (bl[6]=m)) or ((bl[6]=m) and (bl[6]=m)) and (bl[6]=m)								-						
34 if (((b1[0]=jm) and (b1[1]=jm) and (b1[2]=jm) 35 and (b1[7]=jm) and (b1[8]=jm)) or ((b1[0] 36 and (b1[7]=jm) or ((b1[2]=jm) and (b1[5] 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=														
35 and (b1[7]=jm) and (b1[8]=jm)) or ((b1[0] col (1) doblot 3 - integer value 10 to 14 5 to 11 36 and (b1[7]=jm)) or ((b1[2]=jm) and (b1[5] or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]= to array (2) copy 1 [02] integer refarray 18 to 21 12 to 20 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]= to array (2) copy 2 [02] integer refarray 18 to 21 12 to 20							· ·							
36 and (b1[7]=jm)) or ((b1[2]=jm) and (b1[5]) fmmarray (2) copy 1 [02] integer refarray 18 to 21 12 to 20 37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=) to array (2) copy 2 [02] integer refarray 18 to 21 12 to 20				-										
37 or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]= toarray (2)copy 2 [02] integer refarray 18 to 21 12 to 20							-							
							· ·							
	37	or ((b1[2]=jm) and (b1[4]=jm) and (b1[6]=	~	toarray			2	[02]	intege					
1 Canvas QuickHelp 1 QuickHelp 2 Output Syntax Vars/Subs PCode Memory	,	1												

Runs a virtual machine, so keen students can explore 'under the bonnet' without any risk to your computer or network • Teach computer science concepts by inspecting 'machine code' and memory

