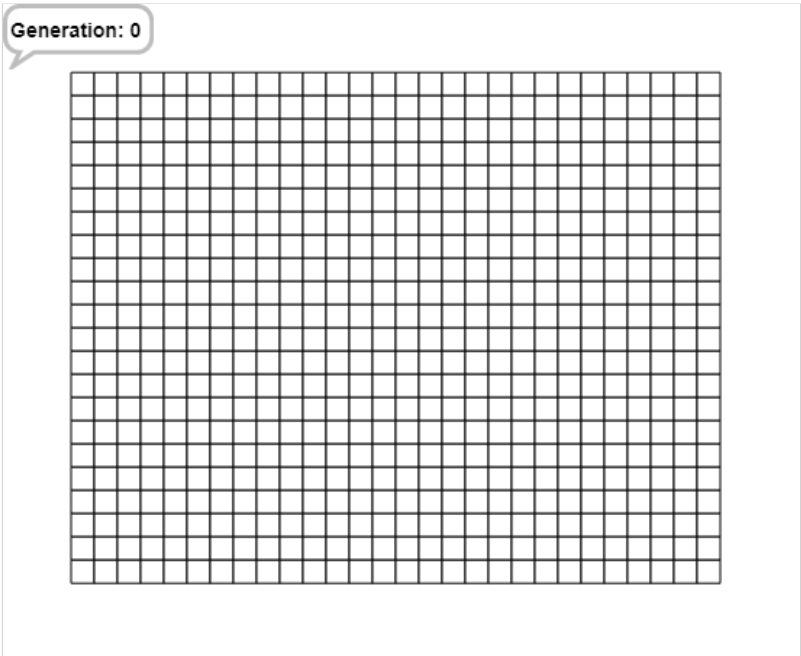
CGOL_student_name

https://snap.berkeley.edu/snap/snap.html#present:Username=tliu6&ProjectName=CGOL_student_name



Snap! 6, https://snap.berkeley.edu

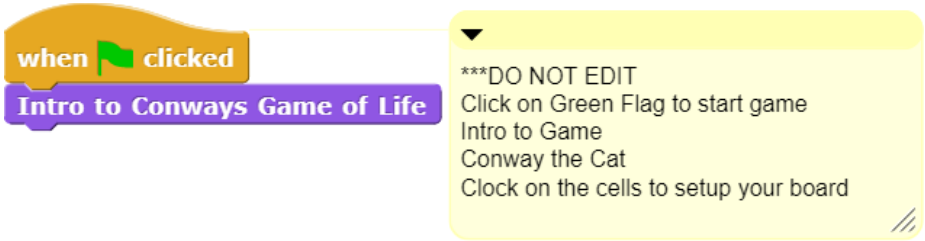
Contents

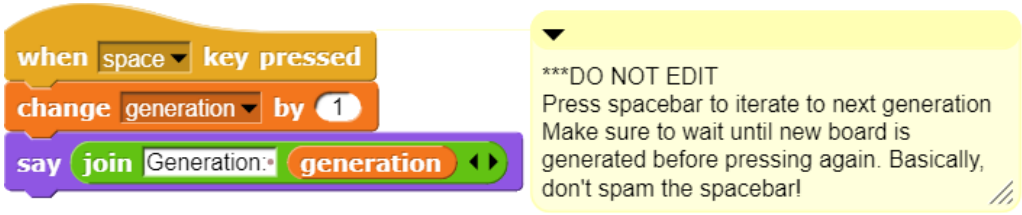
Conway

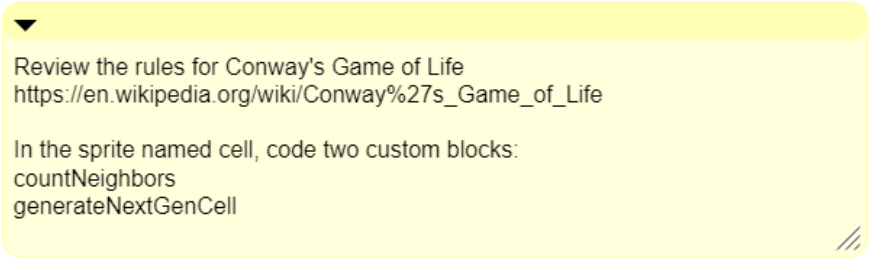
Costumes

1. cat3 
2. blank

Scripts







---

cell

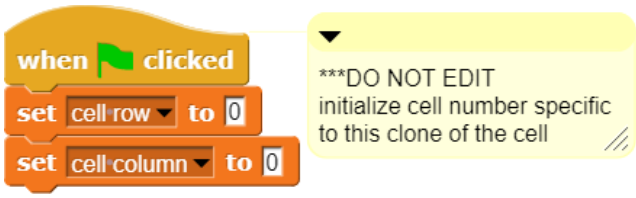
Costumes

1. white cell 
2. black cell 

Variables

- cell cell column 28
- cell cell row 22

Scripts



when cell count > 1500
stop all

***DO NOT EDIT
failsafe. stop program in case clones get out of control and crash the program
If your program is doing nothing you most likely have an error

when I receive draw board
DrawBoard

***DO NOT EDIT
draw initial board after Conway the Cat speaks

when I am clicked
next costume
updateInitialUserSetupCell

***DO NOT EDIT
Changes the cell selection (white/black) and updates the board

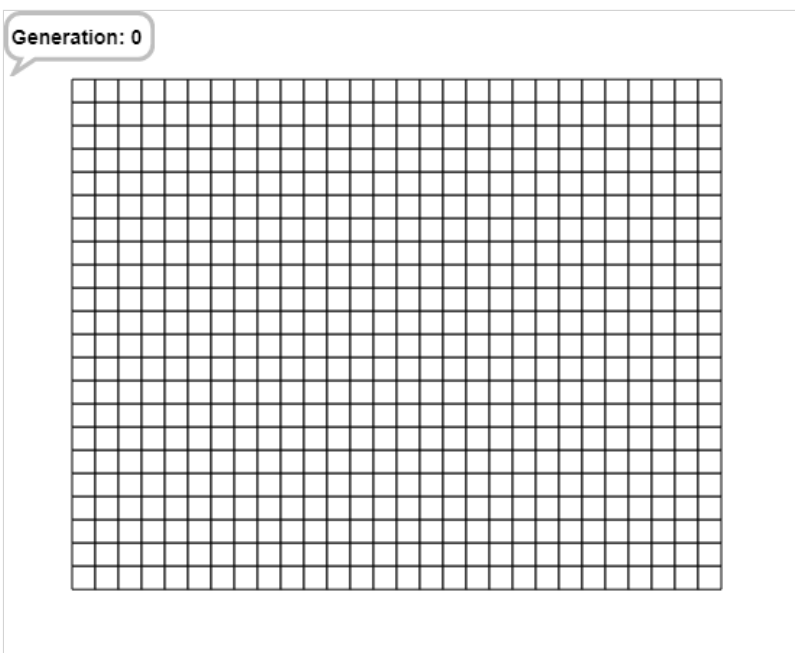
countNeighbors current row current column

create/edit the code in this block to return number of neighbors

when space key pressed
generateNextGenCell
delete this clone

Code the algorithm for Conway's Game of Life INSIDE generateNextGenCell block.
The algorithm to determine if the current cell will be alive or dead
Pressing spacebar will generate the next cell and save into newBoard
No need to modify these 3 blocks of code here.

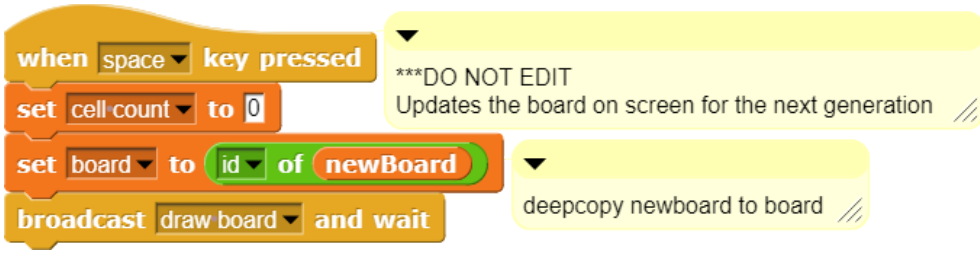
Board



Scripts

when clicked
initialize Board

***DO NOT EDIT
initializes the board



For all Sprites

Variables

- board

22	A	B
1	0	0
2	0	0
3	0	0
4	0	0
	0	0

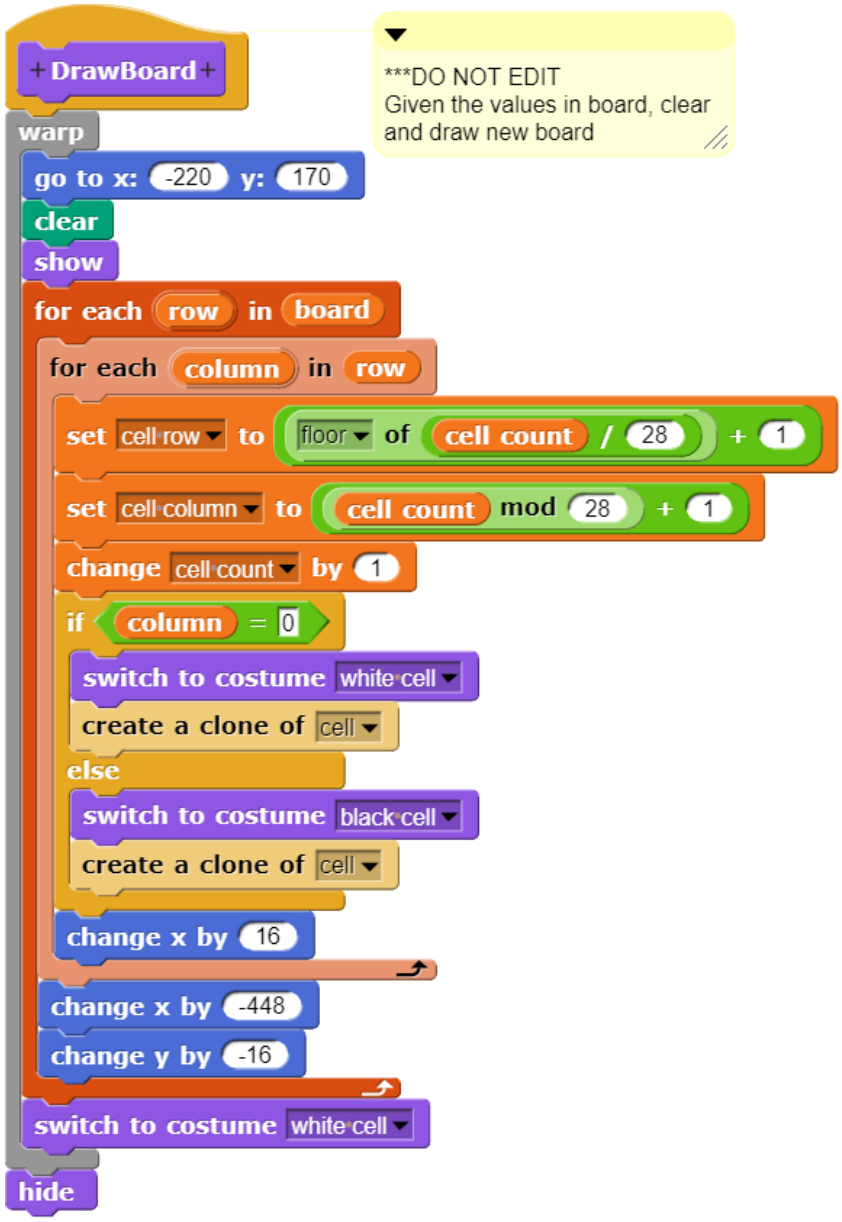
- cell count 616
- generation 0
- newBoard

22	A	B
1	0	0
2	0	0
3	0	0
4	0	0
	0	0

Blocks

Looks

- DrawBoard



- Intro to Conways Game of Life

```
+Intro+to+Conways+Game+of+Life+
go to center
clear
set generation to 0
set cell count to 0
switch to costume cat3
show
say Welcome to Conway's Game of Life for 2 secs
say Click on the cells to initialize the board. Press spacebar to iterate to next generation! for 2 secs
broadcast draw board
switch to costume blank
go to x: -260 y: 400
say join Generation: generation
```

***DO NOT EDIT

- updateInitialUserSetupCell

```
+updateInitialUserSetupCell+
if costume # = 1
  replace item cell column of item cell row of board with 0
else
  replace item cell column of item cell row of board with 1
```

***DO NOT EDIT
Initial board setup, costume indicates alive/dead.

- generateNextGenCell

```
+generateNextGenCell+
```

Implement Conway's Game of Life Algorithm here. Feel free to use the helper blocks
isCellAlive
countNeighbors(which you wrote)
setNextGenCell

Note: You have access to each cell's row and column, see variables list

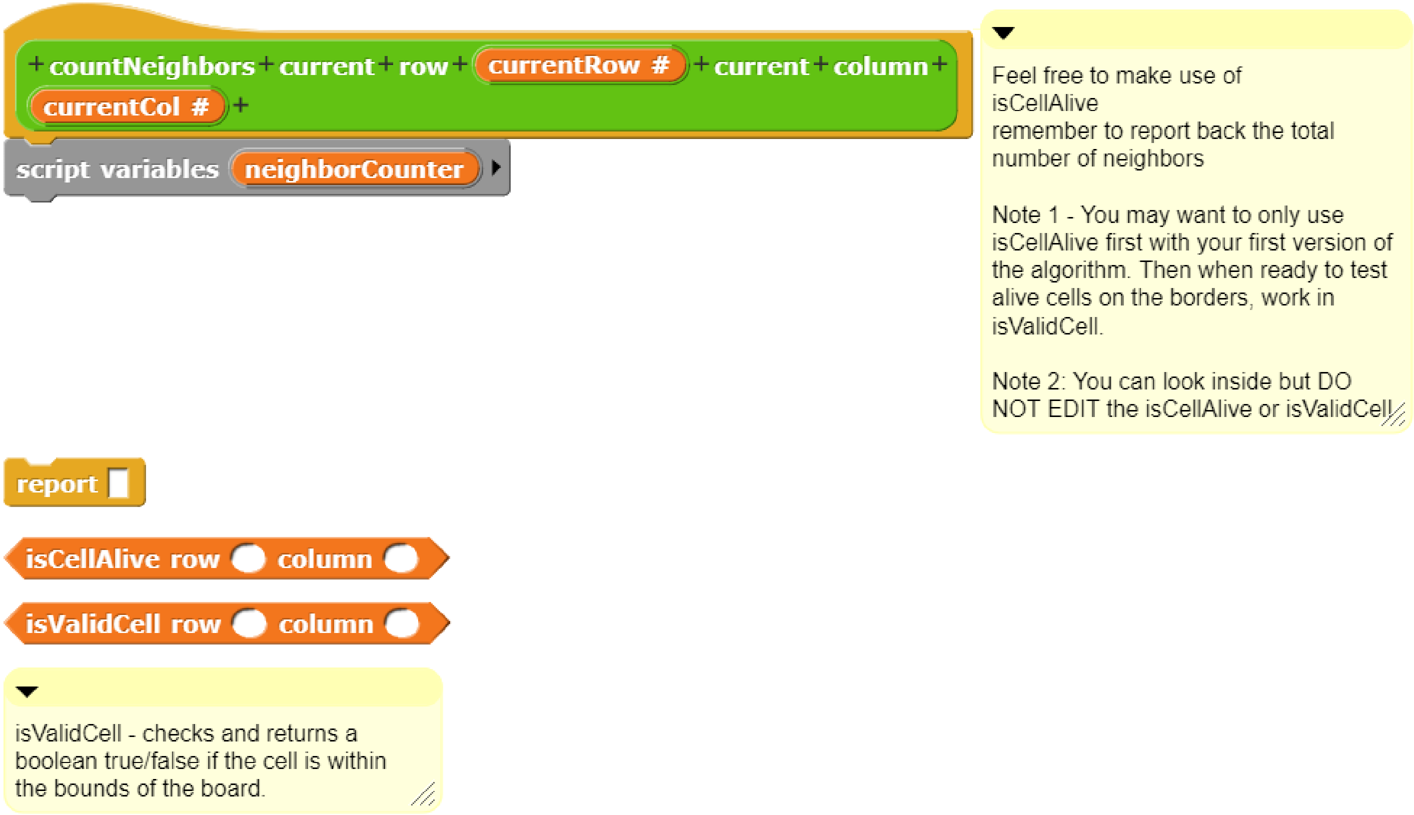
```
isCellAlive row ◯ column ◯
countNeighbors current row ◯ current column ◯
```

setNextGenCell alive ○ row ○ column ○

Operators

- 

countNeighbors current row ○ current column ○



+countNeighbors+current+row+ currentRow # +current+column+ currentCol # +

script variables neighborCounter ▸

report ☐

isCellAlive row ○ column ○

isValidCell row ○ column ○

Feel free to make use of isCellAlive
remember to report back the total number of neighbors

Note 1 - You may want to only use isCellAlive first with your first version of the algorithm. Then when ready to test alive cells on the borders, work in isValidCell.

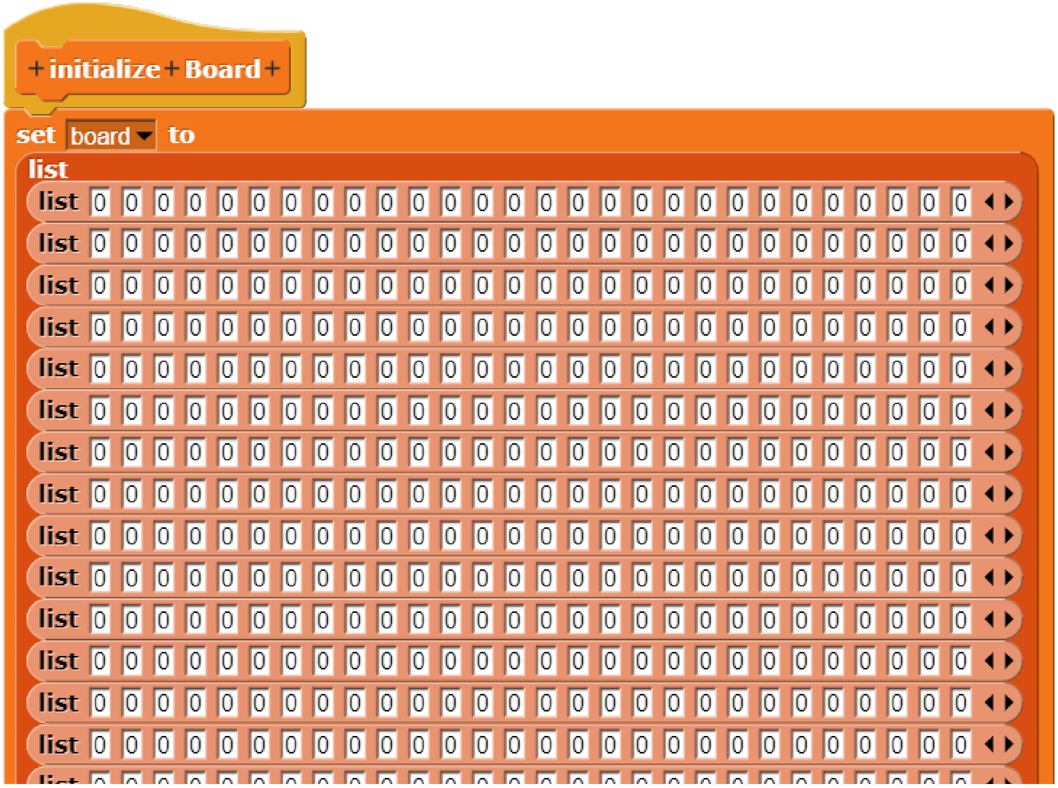
Note 2: You can look inside but DO NOT EDIT the isCellAlive or isValidCell

isValidCell - checks and returns a boolean true/false if the cell is within the bounds of the board.

Variables

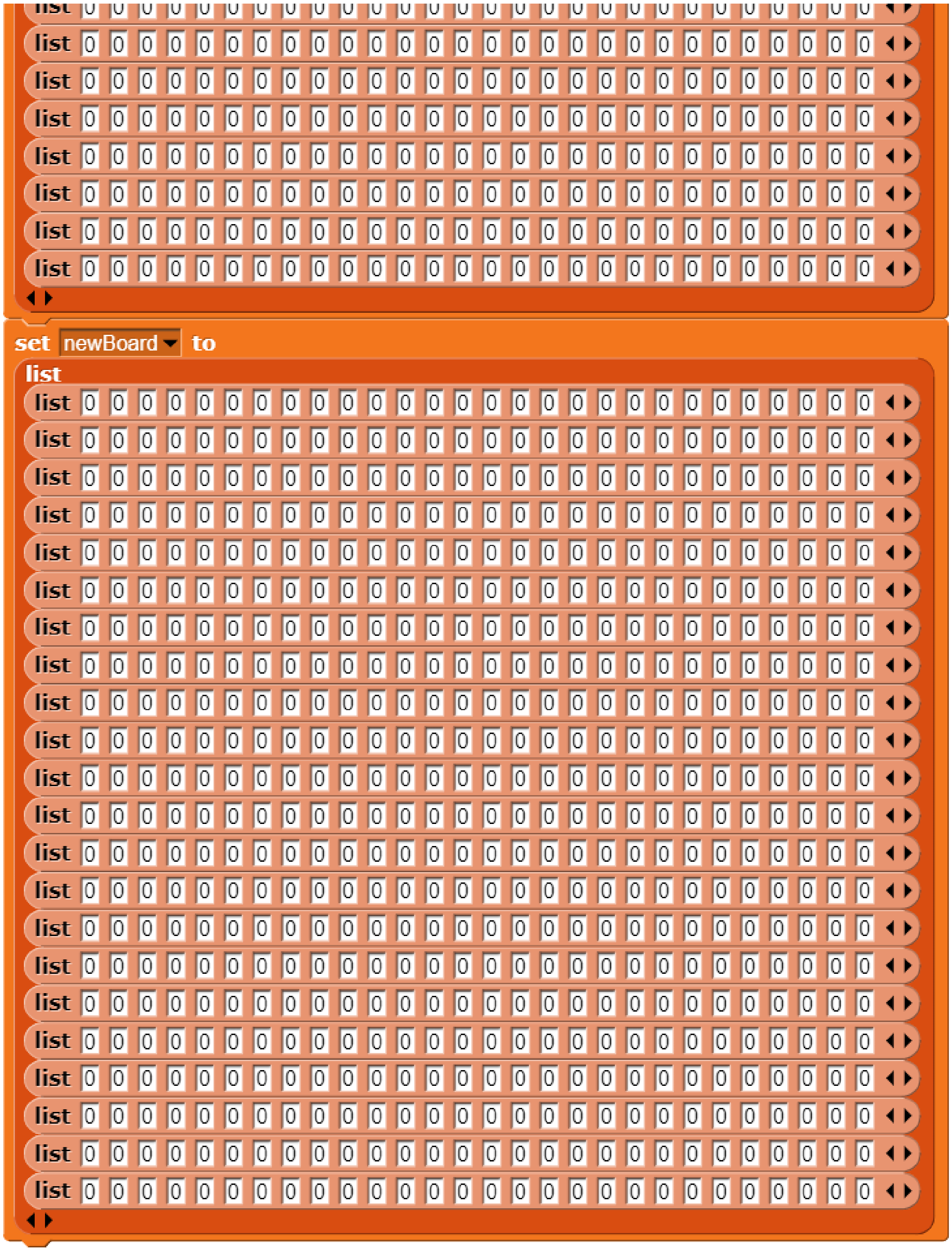
- 

initialize Board

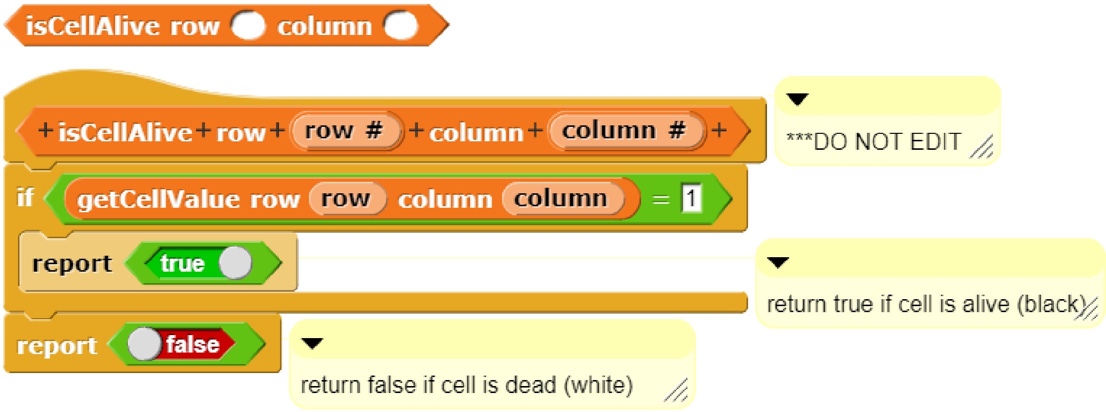


+initialize+Board+

set board to
list
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸



```
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
◂▸
set newBoard to
list
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
  list 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ◂▸
◂▸
```

- 

```
isCellAlive row ( ) column ( )

+isCellAlive+row+(row #)+column+(column #)+      ***DO NOT EDIT
if (getCellValue row (row) column (column)) = 1
    report true                                  return true if cell is alive (black)
report false                                     return false if cell is dead (white)
```

```
report (getCellValue row (row) column (column)) = (1)
```

alternate version can just report or return back the predicate since it's already a Boolean true/false

- isValidCell row ( ) column ( )

```
+isValidCell+row+(row #)+column+(column #)+
if ((row) > (0) and (row) < (23)) and ((column) > (0) and (column) < (29))
    report (true)
report (false)
```

***DO NOT EDIT
Predicate - returns true if row and column reference a valid cell. In other words, checks if the row and column are out of bounds
if row or column <= 0 or
row > 22 or
column > 28.

```
report ((row) > (0) and (row) < (23)) and ((column) > (0) and (column) < (29))
```

alternate version can skip the if/else and just report the Boolean true/false value directly

- setNextGenCell alive ( ) row ( ) column ( )

```
+setNextGenCell+alive+(alive #)+row+(row #)+column+(column #)+
if (alive) = (1)
    replace item (column) of (item (row) of (newBoard)) with (1)
else
    replace item (column) of (item (row) of (newBoard)) with (0)
```

***DO NOT EDIT
sets the cell to be alive or dead in new board for next generation

- getCellValue row ( ) column ( )

```
+getCellValue+row+(row #)+column+(column #)+
report item (column) of (item (row) of (board))
```

*** DO NOT EDIT
Returns value in cell