UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

COMPUTER SCIENCE 1A **DESIGN**

Problem Description

Cellular automata are used in many fields of computer science, physics, theoretical biology etc. One of the well-known cellular automata is known as the Rule 30 elementary cellular automaton. This version of the cellular automaton uses an array of elements that are considered to be either alive or dead. The array evolves into a new version of the array by applying the following rules for each element in the array:

Input & Output;

Input	
Input Description	*Mechanism*
Number of Rounds	Standard Input Stream
Options	Standard Input Stream
Array Values	Standard Input Stream
Output	
Output Description	*Stream (optional)*
Random array values	Standard Output Stream
Generation rounds	Standard Output Stream
Error messages	Standard error stream

Data Format

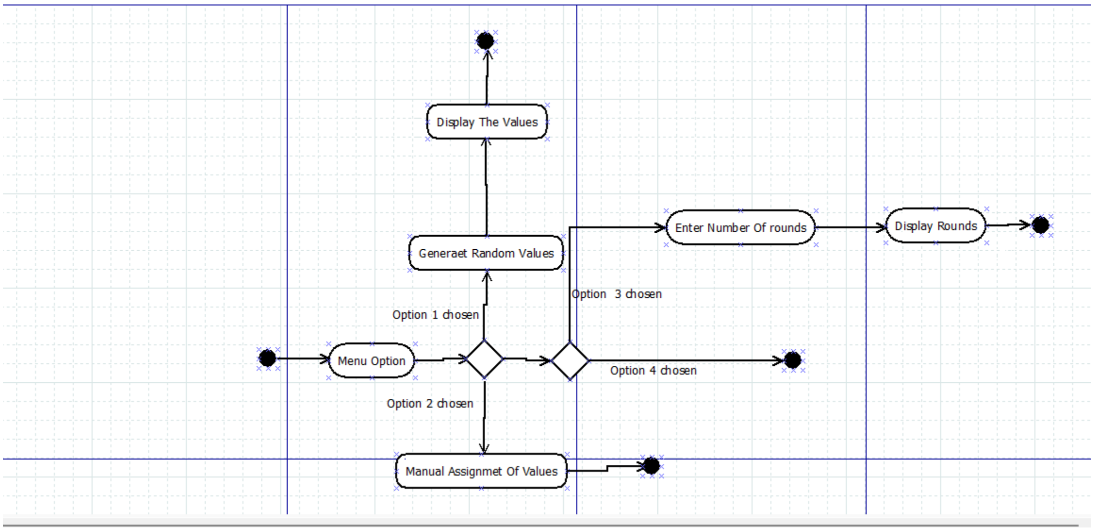
Identifier	*Data Type*	*Description*
nRouunds	integer	For entering number of rounds
ArraySize	integer	For setting the array size
cellsArray	Integer ptr	For taking in array values

Pseudo Code

```
InputNumberOfRounds(nRounds):
   // Take in number of rounds from user
   read nRounds from input

   // Return number of rounds
   return nRounds

RandomArrayValues(cells, arraySize):
   // Loop through the array
   for i from 0 to arraySize - 1:
      // Assign random 0's and 1's
      cells[i] = random number between 0 and 1

      // Display the random values
      output cells[i] with formatting

ManualArrayValues(cells, arraySize):
   // Manual input of 0s and 1s
   output "Please enter " + arraySize + " values of zeros and ones separated by
spaces:"

   // Loop through the array
   for i from 0 to arraySize - 1:
      // Read input value for current cell
      read cells[i] from input

      // Check if input value is out of bounds
      if cells[i] is not 0 and cells[i] is not 1:
         output "One or more values out of bounds. Please enter 1s and 0s only"

DisplayCellRounds(nRounds, arraySize, cells):
   // Round loop
   for r from 1 to nRounds:
      output "Round: " + r

      // Values loop
      for i from 0 to arraySize - 1:
         // Handle edge cases for neighbors
         leftNeighbor = (i == 0) ? cells[arraySize - 1] : cells[i - 1]
         rightNeighbor = (i == arraySize - 1) ? cells[0] : cells[i + 1]
```

```
        // Apply the rules of the cellular automaton to update the current cell
        if leftNeighbor is 1 and cells[i] is 1 and rightNeighbor is 1:
            cells[i] = 0
        else if leftNeighbor is 1 and cells[i] is 1 and rightNeighbor is 0:
            cells[i] = 0
        else if leftNeighbor is 1 and cells[i] is 0 and rightNeighbor is 1:
            cells[i] = 0
        else if leftNeighbor is 1 and cells[i] is 0 and rightNeighbor is 0:
            cells[i] = 1
        else if leftNeighbor is 0 and cells[i] is 1 and rightNeighbor is 1:
            cells[i] = 1
        else if leftNeighbor is 0 and cells[i] is 1 and rightNeighbor is 0:
            cells[i] = 1
        else if leftNeighbor is 0 and cells[i] is 0 and rightNeighbor is 1:
            cells[i] = 1
        else if leftNeighbor is 0 and cells[i] is 0 and rightNeighbor is 0:
            cells[i] = 0

        // Output the updated value of the cell with formatting
        output cells[i] with formatting

    // End of line after displaying the values of the round
    output end of line
```

UML Activity Diagram


Display The Values
Enter Number Of rounds
Display Rounds
Generaet Random Values
Option 3 chosen
Option 1 chosen
Menu Option
Option 4 chosen
Option 2 chosen
Manual Assignmet Of Values