Assignment

Imagine an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent.

At each step in time (tick), the following transitions occur:

1. Any live cell with fewer than two live neighbors dies, as if by loneliness.

2. Any live cell with more than three live neighbors dies, as if by overcrowding.

3. Any live cell with two or three live neighbors lives, unchanged, to the next generation.

4. Any dead cell with exactly three live neighbors comes to life.

The initial pattern constitutes the 'seed' (randomly placed 500 cells) of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed — births and deaths happen simultaneously, and the discrete moment at which this happens is called a tick. (In other words, each generation is a pure function of the one before.)

Program:-

```
package capitaprep;

import java.util.*;

public class Coditation

{

  public static void main(String args[])

    {

        Coditation  s=new Coditation ();
```

```
  Scanner sc=new Scanner(System.in);
    int r=10,c=10;
   char choice;
   int ch;
  // Initialise the grid
   int[][] grid= { { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
     { 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
     { 0, 0, 0, 0, 1, 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, 1, 1, 0, 0, 0, 0, 0 },
     { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0 },
     { 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 },
     { 0, 0, 0, 0, 1, 0, 0, 0, 0, 0 },
     { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
   };

  int grid1[][] = new int[r][c];

do
  {

 System.out.println("======Game of life======");
System.out.print("\nThis is the original generation:\n");
for(int i=0;i<grid.length;i++)
```

```
            {
            for(int j=0;j<grid.length;j++)
            {
                  System.out.print(grid[i][j]+" ");
            }
            System.out.println();
            }
      System.out.println("=====Menu=====");
       System.out.println("\n1.Next generation");
      System.out.println("\n2.Cell status");
       System.out.println("\nEnter Your Choice ");
       ch=sc.nextInt();

       switch(ch)
       {
          case 1 : System.out.println("The next generation is:\n");
                s.nextGrid(grid,r,c,grid1);
                  break;

          case 2 :
                 s.checkStatus(grid1);
                  break;
```

```
            default:
                    System.out.println("Wrong Choice");
        }
         System.out.println("Do you want to continue:Press(y || n)");
           choice=sc.next().charAt(0);
        }while(choice=='y');
    }
public  void nextGrid(int grid[][], int r, int c,int grid1[][])
     {
       for (int l = 1; l < r - 1; l++) // Looping through every cell
          {
           for (int m = 1; m < c - 1; m++)
           {
              int alive_Neighbours = 0; // finding no Of Neighbours that are alive
              for (int i = -1; i <= 1; i++)
                  for (int j = -1; j <= 1; j++)
              alive_Neighbours += grid[l + i][m + j];



              if ((grid[l][m] == 1) && (alive_Neighbours < 2)) // Cell is lonely and dies
                  grid1[l][m] = 0;
```

```
                else if ((grid[l][m] == 1) && (alive_Neighbours > 3))
// Cell dies due to overcrowding
                    grid1[l][m] = 0;



               else if ((grid[l][m] == 0) && (alive_Neighbours == 3))
// A new cell is born
                    grid1[l][m] = 1;
                else
                    grid1[l][m]=grid[l][m];
// Remains the same
            }


        }


       for (int i = 0; i < r; i++)
       {
           for (int j = 0; j < c; j++)
           {
              System.out.print(grid1[i][j] + " ");
           }
           System.out.println();
        }


   }
```

```
public  void checkStatus(int grid1[][])
     {
       Scanner sc=new Scanner(System.in);
         System.out.println("==============Enter  cell  you want to Check:==========");
         System.out.println("Enter row you want to Check:");
          int                                        cell1=sc.nextInt(); //Input

         System.out.println("Enter col you want to Check ");
          int                                        cell2=sc.nextInt(); //Input

          if(grid1[cell1][cell2]==1)
              System.out.println("cell status is live.");
             else
              System.out.println("cell status is dead.");
      }
   }
```

Output:-

```
run:
=====Game of life======

This is the original generation:
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 1 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 1 1 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
=====Menu=====

1.Next generation

2.Cell status

Enter Your Choice
```

1st choice output:-

```
Enter Your Choice
1
The next generation is:

0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 1 1 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 1 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Do you want to continue:Press(y || n)
y
```

2nd choice output:-

```
Do you want to continue:Press(y || n)
y
=====Game of life======

This is the original generation:
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 0 1 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 1 1 0 0 0 0 0
0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
=====Menu=====

1.Next generation

2.Cell status

Enter Your Choice
2
==============Enter cell you want to Check:===========
Enter row you want to Check:
5
Enter col you want to Check
3
cell status is dead.
Do you want to continue:Press(y || n)
n
BUILD SUCCESSFUL (total time: 19 seconds)
```