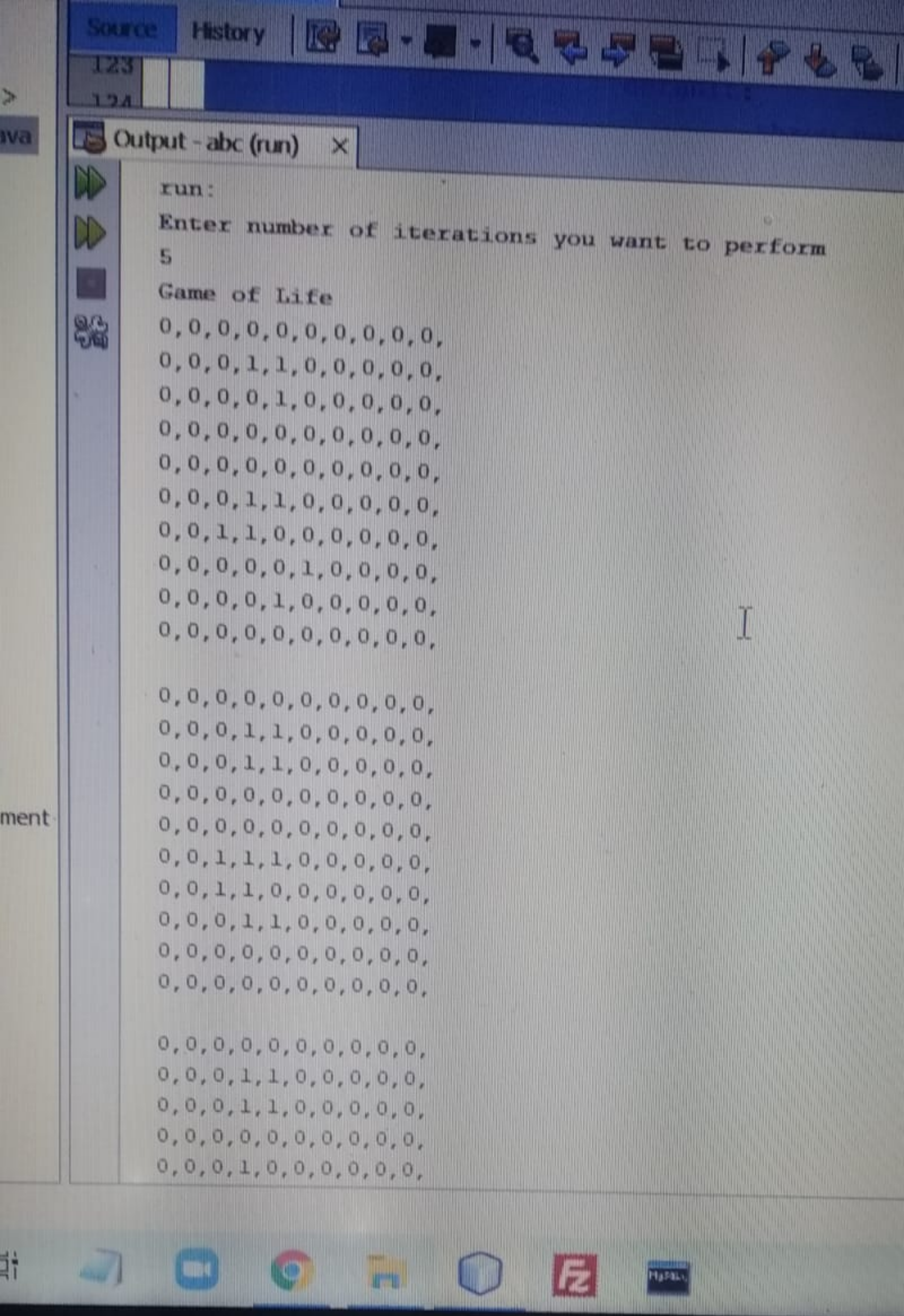
```
Coding:
================================================================================
=========

import java.util.Scanner;

public final class GameOfLife
{
      public final static int DEAD=0x00;
      public final static int LIVE=0x01;

      private void test(int itn)
        {
            int[][] board={
                    { 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 }
                              };
            System.out.println("Game of Life");
            printBoard(board);

            for (int i=0; i<itn; i++)
                {
                  System.out.println();
                  board=getNextBoard(board);
                  printBoard(board);
            }

            Scanner sc= new Scanner (System.in);
            System.out.println("Enter The Cell  You Want To Check ");

            System.out.println("Enter Row ");
            int r=sc.nextInt();

            System.out.println("Enter Column ");
            int c=sc.nextInt();
            if(r<board.length && c<board[0].length)
                {
                  int count=0;
                  for (int i=0; i<board.length; i++)
                        {
                        for (int j=0; j<board[i].length; j++)
                                {
                              if (board[r][c]==0)
                                    count=0;
                              else
                                    count=1;
                        }

                  }
                  if (count==0)
                        System.out.println("*********Cell is Dead*********");
                  else
                        System.out.println("*********Cell is Live*********");
            }
      }
```

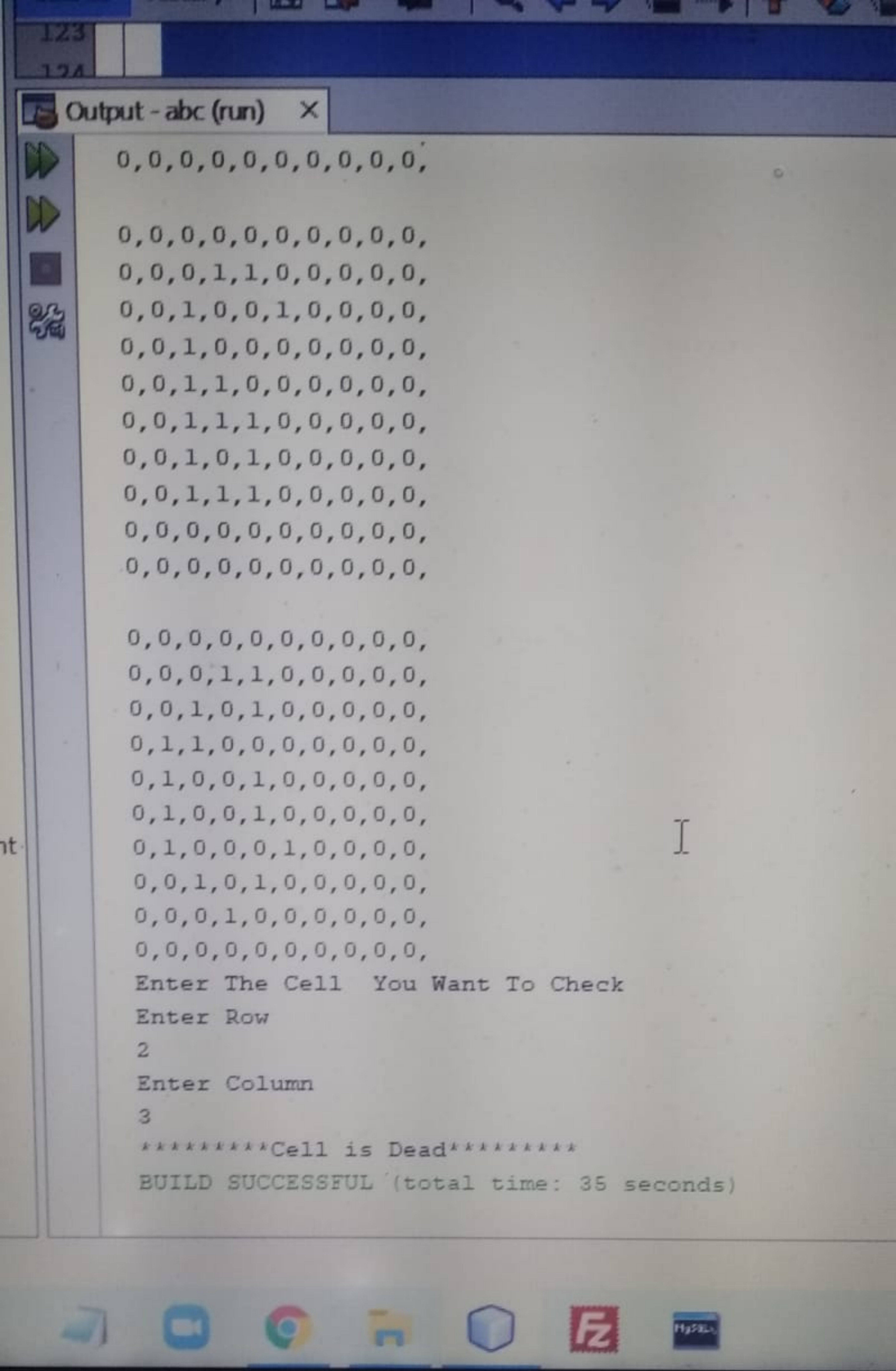
```
       private void printBoard(int[][] board)
         {
             for (int i=0, x=board.length; i<x; i++)
                 {
                   for (int j=0,y=board[i].length; j<y; j++)
                         {
                    System.out.print(Integer.toString(board[i][j])+",");
                   }
                    System.out.println();
             }

       }

       public int[][] getNextBoard(int[][] board)
         {
             if(board.length==0 || board[0].length==0)
                  throw new IllegalArgumentException("Board must have a positive
amount of rows and/or columns");

             int nRows=board.length;
             int nCols=board[0].length;
             int [][] temp= new int[nRows][nCols];
             for (int row= 0; row<nRows; row++)
                 {
                   for (int col=0; col<nCols; col++)
                         {
                    temp[row][col]=getNewCellState(board[row]
[col],getLiveNeighbour(row,col,board));
                   }
             }
             return temp;
       }
       private int getLiveNeighbour(int cellRow, int cellCol, int[][]board)
         {
             int liveNeighbours=0;
             int rowEnd=Math.min(board.length, cellRow+2);
             int colEnd=Math.min(board[0].length, cellCol+2);

             for (int row=Math.max(0, cellRow-1); row<rowEnd; row++)
                 {
                   for (int col=Math.max(0, cellCol-1); col<colEnd; col++)
                         {
                    if((row!=cellRow || col!=cellCol) && board[row][col]==
LIVE)liveNeighbours++;
                   }
             }
             return liveNeighbours;
       }

       private int getNewCellState(int curState, int liveNeighbours)
         {
             int newState= curState;
             switch (curState){
             case LIVE:
                   if(liveNeighbours<2)
                         newState=DEAD;
                   if(liveNeighbours==2 || liveNeighbours==3)
                         newState=LIVE;
                   if(liveNeighbours>3)
                         newState=DEAD;
                   break;
             case DEAD:
                   if(liveNeighbours==3)
                         newState=LIVE;
                   break;
                   default:
                         throw new IllegalArgumentException("State of cell must
be either LIVE or DEAD");

             }
             return newState;
       }
      public final static void main(String[] args)
        {
            GameOfLife gol=new GameOfLife();
            Scanner sc= new Scanner (System.in);
            System.out.println("Enter number of iterations you want to
perform");
            int num=sc.nextInt();
            gol.test(num);
      }

}
```

```
================================================================================
================================
Output:
================================================================================
================================

Enter number of iterations you want to perform
5
Game of Life
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,

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,
0,0,1,1,1,0,0,0,0,0,
0,0,1,1,0,0,0,0,0,0,
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,

0,0,0,0,0,0,0,0,0,0,
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,
0,0,0,1,0,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,1,1,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,0,0,
0,0,0,1,1,0,0,0,0,0,
0,0,0,1,1,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,
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,
0,0,0,1,1,0,0,0,0,0,
0,0,1,0,0,1,0,0,0,0,
0,0,1,0,0,0,0,0,0,0,
0,0,1,1,0,0,0,0,0,0,
0,0,1,1,1,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,1,1,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,1,1,0,0,0,0,0,0,0,
0,1,0,0,1,0,0,0,0,0,
0,1,0,0,1,0,0,0,0,0,
0,1,0,0,0,1,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
Enter The Cell  You Want To Check
Enter Row
2
Enter Column
3
*********Cell is Dead*********
```



```
run:
Enter number of iterations you want to perform
5
Game of Life
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,

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,
0,0,1,1,1,0,0,0,0,0,
0,0,1,1,0,0,0,0,0,0,
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,

0,0,0,0,0,0,0,0,0,0,
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,
0,0,0,1,0,0,0,0,0,0,
```

2021-9-18 20:39



```
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,
0,0,1,0,0,1,0,0,0,0,
0,0,1,0,0,0,0,0,0,0,
0,0,1,1,0,0,0,0,0,0,
0,0,1,1,1,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,1,1,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,1,1,0,0,0,0,0,0,0,
0,1,0,0,1,0,0,0,0,0,
0,1,0,0,1,0,0,0,0,0,
0,1,0,0,0,1,0,0,0,0,
0,0,1,0,1,0,0,0,0,0,
0,0,0,1,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
Enter The Cell  You Want To Check
Enter Row
2
Enter Column
3
*********Cell is Dead*********
BUILD SUCCESSFUL (total time: 35 seconds)
```