Green Vs. Red

# Introduction

Green vs. Red is a game played on a 2D grid that in theory can be infinite (in our case we will assume that x <= y < 1000).

Each cell on this grid can be either green (represented by 1) or red (represented by 0). The game always receives an initial state of the grid which we will call ‘Generation Zero’. After that a set of 4 rules are applied across the grid and those rules form the next generation.

Rules that create the next generation:

1. Each red cell that is surrounded by exactly 3 or exactly 6 green cells will also become green in the next generation.
2. A red cell will stay red in the next generation if it has either 0, 1, 2, 4, 5, 7 or 8 green neighbours.
3. Each green cell surrounded by 0, 1, 4 ,5 ,7 or 8 green neighbours will become red in the next generation.
4. A green cell will stay green in the next generation if it either 2, 3 or 6 green neighbours.

Important facts:

* Each cell can be surrounded by up to 8 cells 4 on the sides and 4 on the corners. Exceptions are the corners and the sides of the grid.
* All the 4 rules apply at the same time for the whole grid in order for next generation to be formed.

**Your task:**

Create a program that accepts:

The size of our grid –x, y (X being the width and y being the height).

Then the next y lines should contain strings (long x characters) created by 0s and 1s which will represent the Generation Zero stat and help us build the grid.

The last arguments to the program should be coordinates (x1 and y1) and the number N.

(x1 and y1) will be coordinates of a cell in the grid. We would like to calculate in how many generations from Generation Zero until generation N this cell was green. (The calculation should include generation Zero and generation N)

Print your result in the console.

Special requirement: Write your game in a way that uses several classes. This will show OOP knowledge and will account for more points during the evaluation. Comments, good naming convention and documentation are also recommended.

Example1:

# 3x3 grid, in the initial state, the second row is all 1s, how many times, will the cell [1, 0] (top center) become green in 10 turns?

3, 3

000

111

000

1, 0, 10

# expected result: 5

Example2:

# 4x4 grid. Input:

4, 4

1001

1111

0100

1010

2, 2, 15

# expected result: 14

# Classes with explanation what are their functionality

# Main - class

In the main has the method which has the logic for green’s in one cell, it made in main because we can turn it for counting the null’s too.

# GetNeighbours – class

In that class we calculate how many neighbours do have a current cell, which we are looping.

# Rules – class

In the introduction is told the rules, which we need to keep by thinking our logic.

# NextGenerations – class

In this class we are looping next generations and create them and we get the neighbours and store our grid.

# FindCountOfNeighbours – class

This class contains the count of the neighbours in current cell.

# PrintGrid – class

We just print our grid and update after each loop.

# Created by

Owner Seyhan Mustafov

Email – [seyhan.mustafov@hotmail.de](mailto:seyhan.mustafov@hotmail.de)

Contents

[Introduction 1](#_Toc46711235)

[Classes with explanation what are their functionality 2](#_Toc46711236)

[Main - class 2](#_Toc46711237)

[GetNeighbours – class 3](#_Toc46711238)

[Rules – class 3](#_Toc46711239)

[NextGenerations – class 3](#_Toc46711240)

[FindCountOfNeighbours – class 3](#_Toc46711241)

[PrintGrid – class 3](#_Toc46711242)

[Created by 3](#_Toc46711243)