COMP281 2019 – Assignment 2

- In the following, you will find the problems that constitute Assignment 1. They will be also available on the *online judging (OJ)* system available at https://student.csc.liv.ac.uk/JudgeOnline

- You need to write a C program (not C++ or C#) that solves each problem – it must read the input, as specified in the problem description then print the solution to the given problem for that input.
  - Note that code is "correct" only if it **correctly implements a solution to the problem stated** in the assignment, not "if online judge accepts it".
  - That is, even if OJ accepts your code, it could be wrong. Read the problems carefully.

- Input is read from the standard input, in the same way that you read input from the keyboard as shown in lectures (e.g., using scanf). Output is also printed to the standard output, as you have seen (e.g., using printf).

- When you are satisfied that your C programs work correctly, you must submit them through the departmental submission system.
  - Even if the program is not correct, still submit whatever you have! You can still earn points if certain parts of the code are done right.

- You must also include a brief report describing your solutions to the problems. This should be maximum two sides of A4 paper and should give a description of how each of your solutions works. This should include describing the algorithm used to reach the solution, describing your use of any C language features (that were not discussed in lectures) and identifying any resources that you have used to help you solve the problems.

- This assignment is worth 50% of the total mark for COMP281.
  - All problems are weighted equally.
  - For each problem, you can earn a total of 50 points
    - 25 points for "Functionality and Correctness" awarded for programs that **correctly** solve the problem for all test cases.
    - 20 points for "Programming style, use of comments, indentation and identifiers" awarded depending on the style, comments and efficiency of the solution
    - 5 points for the quality and depth of the accompanying report
  - The final grade results from normalising the earned points to a scale of 100.
  - See separate "comp281-detailed-marking-guidelines.pdf" for more details.

Submission Instructions

- Create a folder, and name it using your Student ID and User ID, e.g. 201234567_sgxyz6
- In the folder, there should be 6 files:
  - 1 report file, in PDF format. Name it with your Student ID and User ID, e.g. 201234567_sgxyz6.pdf
  - 5 source code files. Name each using the Problem Number, e.g. 1006.c
    - In your source code, include your Student Info and Problem Info, e.g.:

```
/*
 * Student ID: 201234567
 * Student Name: Bruce Lee
 * Email: bruce.lee@student.liverpool.ac.uk
 *
 * User: sgxyz6
 *
 * Problem ID: 1006
 * RunID: 22456
 * Result: Accepted
 */
```

    - The OJ provides a RunID, which is different from the Problem ID.
    - The Result is one of the following: Accepted, Wrong Answer, Presentation Error, Time Limit Exceeded, Memory Limit Exceeded, Output Limit Exceeded, Runtime Error, Compile Error.
- Compress the folder into a single zip file, and name it as, e.g. 201234567_sgxyz6.zip
  - Use the standard (pkzip) zip file format:
    https://en.wikipedia.org/wiki/Zip_%28file_format%29
    which is supported by winzip, winrar, etc. on Windows/Mac OS X, and 'zip' on Linux
  - Test your zip file before submitting.
- Submit this zip file using the departmental submission system at
  http://www.csc.liv.ac.uk/cgi-bin/submit.pl
  Only the file submitted through this link will be marked.
- The **deadline** for this assignment submission is **08-Mar-2019 17:00**
- Penalties for late submission apply in accordance with departmental policy as set out in the student handbook, which can be found at: http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf

1. 1081

Title: Game of Life

Description

In this assignment, you will code a simulation called "Game of Life". It is a very famous 'game' and the formed the basis of an entire field of simulation models called "cellular automata". Before continuing reading this assignment, I suggest you read a bit more about Game of Life at:

https://en.wikipedia.org/wiki/Conway's_Game_of_Life
or
http://www.math.com/students/wonders/life/life.html
(the latter also has a nice Java applet of the game that is quite fun to play with!)

You should now know about the basics of game of life: it is a 'simulation' of cells that are 'dead' or 'alive' in discrete time steps, at every step all of the cells will get computed a new value in parallel (i.e., based on the values of the neighbours at the *previous* time step - this will require a little bit of thinking!), and the rules to determine if a cell is dead or alive are:

-Any live cell with fewer than two live neighbours die, as if caused by under population.
-Any live cell with two or three live neighbours lives on to the next generation.
-Any live cell with more than three live neighbours dies, as if by overpopulation.
-Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

You now need to write a program that reads in an initial 'board' configuration, and then performs a specified number of simulations.

As always, try and keep your program clean, modular, and only allocate as much memory as you need (using malloc).

Input

The first line will specify 3 numbers:

-the number of rows
-the number of columns
-the number of steps you need to simulate

The remainder of the input file comprises the initial configuration, which will have exactly the specified number of rows and columns. Dead cells are indicated with '.' while live cells are marked with 'X'.

Output

As output, you will need to write the board configuration that results from the specified number of simulations, in exactly the same format as you read the input.

Sample Input

```
6 6 20
.X...X
X.X.X.
X...X.
X..XX.
..XX.X
...X.X
```

Sample Output

```
...X..
..X.X.
..X.X.
...X..
......
......
```

2. Problem 1084

Title: Highway Lite

Description

Note: this is a simplified version of "Highway"

=Overview=
In this problem you are asked to implement a simplistic highway simulation. The 'highway' is (rows x cols) grid of cells that can be occupied by a vehicle or not. All traffic starts at the left side of of the grid (at column 0) and travels right on, never switching lane (i.e., row). If a car moves beyond the last column it exits the simulation. As input you will be given a list of arrival times of vehicles, the goal is to produce the state of the highway after a specified number of simulation steps.

=The Highway=
The highway is a (rows x cols) grid, where row 0 corresponds to the topmost lane, and column 0 is the leftmost segment. I.e., a 3 lane highway of 10 segments looks like this:

```
..........    <- row 0, is the topmost lane
..........
..........

^
|
column 0, is the leftmost segment
```

=Vehicles=
In this version, there is just 1 vehicle type: all vehicles get to move 1 step to the right in each time step.

=Movement, Priorities & Arrivals=
The simulation is executed in discrete time steps. Each step consists of 2 phases:

1) the movement phase, and
2) the arrival phase.

In the movement phase, all the vehicles that are already on the highway get to move. They do not move synchronous, but one after another: vehicles that entered the simulation first get to move first.

After all vehicles already on the board have been given the opportunity to move, new vehicles can arrive. These are read from a list that serves as the program input.

=Detailed 1 lane Example=
Here we provide a detailed example of how the simulation should be executed. Suppose this is the input:

```
1 50 40
2 0
5 0
9 0
11 0
14 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
28 0
30 0
32 0
34 0
36 0
38 0
```

Then the following simulation should occur:

(Note, the line after the indicated time step shows the state at the beginning of that time step. So, the first car arrives in the arrival phase of timestep t=2 and therefore is shown at the beginning of time step t=3)

```
t=0
................................................
t=1
................................................
t=2
................................................
t=3
1...............................................
t=4
.1..............................................
t=5
..1.............................................
t=6
1..1............................................
t=7
.1..1...........................................
t=8
..1..1..........................................
t=9
...1..1.........................................
t=10
1...1..1........................................
t=11
.1...1..1.......................................
t=12
1.1...1..1......................................
t=13
.1.1...1..1.....................................
t=14
..1.1...1..1....................................
t=15
1..1.1...1..1...................................
t=16
.1..1.1...1..1..................................
t=17
..1..1.1...1..1.................................
t=18
...1..1.1...1..1................................
t=19
....1..1.1...1..1...............................
t=20
.....1..1.1...1..1..............................
t=21
1.....1..1.1...1..1.............................
t=22
11.....1..1.1...1..1............................
t=23
111.....1..1.1...1..1...........................
t=24
1111.....1..1.1...1..1..........................
t=25
11111.....1..1.1...1..1.........................
t=26
111111.....1..1.1...1..1........................
t=27
1111111.....1..1.1...1..1.......................
t=28
.1111111.....1..1.1...1..1......................
t=29
1.1111111.....1..1.1...1..1.....................
t=30
.1.1111111.....1..1.1...1..1....................
t=31
1.1.1111111.....1..1.1...1..1...................
t=32
.1.1.1111111.....1..1.1...1..1..................
t=33
1.1.1.1111111.....1..1.1...1..1.................
```

```
t=34
.1.1.1.1111111.....1..1.1...1..1.................
t=35
1.1.1.1.1111111.....1..1.1...1..1.................
t=36
.1.1.1.1.1111111.....1..1.1...1..1...............
t=37
1.1.1.1.1.1111111.....1..1.1...1..1..............
t=38
.1.1.1.1.1.1111111.....1..1.1...1..1.............
t=39
1.1.1.1.1.1.1111111.....1..1.1...1..1............
t=40
.1.1.1.1.1.1.1111111.....1..1.1...1..1...........
```

As such, the requested output in this case would be:

```
.1.1.1.1.1.1.1111111.....1..1.1...1..1...........
```

Input

First a row with 3 integers:

```
number_of_rows number_of_colums number_of_timesteps
```

that specify the size of the highway and the number of steps to simulate.

This is followed by the aforementioned list of arrivals. Each entry is a row with 2 integers:

```
arrival_time row_index
```

-These rows will be ordered based on arrival time.

Output

The state of the highway after number_of_timesteps have been simulated, encoded as follows:

'.' denotes empty space
'1' denotes a vehicle

Sample Input

```
5 10 20
0 1
0 3
0 4
2 4
4 3
4 4
6 3
6 4
10 0
10 1
12 2
12 4
18 1
```

Sample Output

```
.........1
.1.......1
.......1..
..........
.......1..
```