Assignment #8

Concepts to be utilized: *2D dynamic array and File handling*

Submission deadline: 21st May, 2020, Thursday till 11:59pm

Submission guidelines:

- Put all your .cpp files in a folder. Zip that folder and name it with your roll number.
- Email the zipped folder at **natalia@pucit.edu.pk**. You must put the **subject** of your email as: **"A#8"**
- Students who are facing issues with their laptop or compiler can write the code on paper and send the screenshots, following the above guideline. Since your issue will be genuine therefore relaxation will be given accordingly. Don't worry!

Prison challenge — 20 marks

A two-dimensional integer array contains mostly 0's and 1's and a single entry of 2. The 0's represent empty space, the 1's represent wall and the 2 represents the trapped prisoner. You can take the following assumptions:

i. The array is mostly empty space (i.e. lot of 0's)

ii. The array contains a RECTANGLE (Room made of walls) (i.e. a shape made of 1's)

iii. The Room may not be totally closed (walls all around)

iv. The trapped prisoner (represented by 2) may be inside or outside the RECTANGLE

Sample Input: 10x10

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**	**1**	**1**	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	**2**	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	**1**	**1**	**0**	**1**	**1**	0	0
0	0	0	0	0	0	0	0	0	0

Output: Inside prison but can be freed.

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**	**1**	0	0	0	0
0	0	**1**	0	0	**1**	0	0	0	0
0	0	**1**	0	0	**1**	0	0	**2**	0
0	0	**1**	0	0	**1**	0	0	0	0
0	0	**1**	0	0	**1**	0	0	0	0
0	0	**0**	0	0	**1**	0	0	0	0
0	0	**1**	**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

Output: **Outside Room (Already free)**

Sample Input: 10x10

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**	**1**	**1**	**1**	0	0
0	0	**1**	**2**	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	**1**	**1**	**0**	**1**	**1**	0	0
0	0	0	0	0	0	0	0	0	0

Ouput: Inside Prison but breakable.

Sample Input: **10x10**

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**	**1**	**1**	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	**2**	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	0	0	0	0	**1**	0	0
0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
0	0	0	0	0	0	0	0	0	0

Output: Trapped. You can't escape the prison

Implement a function **CheckPrisonerStatus** which takes a two-dimensional array (make sure you allocate a 2 dimensional array using pointers; i.e. dynamic array) as an argument. It should create an output file named "Status.txt" and write the status in it regarding if the prisoner is already freed or it is trapped completely inside a room *(made up of unbreakable walls)* or trapped inside the room which has an open/breakable wall *(walls made up of 1's but that wall has at least one entry of zero).* While initializing a 2D map of wall, you can ask user about the top left and bottom right index of an array map. You can also ask the user if he wants to have breakable wall or not. You can construct rectangular boundary map using this. You also need to take index position of prisoner. You can also add relevant functions to support the logic.

The main function should declare an array and call the appropriate function with that array passed. Since it would be dynamic array therefore you must take row and column size as an input from the user!