Big Homework 1

Data Structures and Algorithms

Stacks & Queues

General mentions

• For this project you will work either in teams of 2 people or alone.

• The homework will be uploaded on the Moodle platform (fils.curs.pub.ro), for Hw1 assignment. If you encounter problems with the platform, contact by email your lab assistant.

• The homework must be submitted by 31.03.2019, at 08:00 AM. No late submissions will be accepted.

• The evaluation of the project will take place during the first lab after the submission deadline. Presence at the lab will be mandatory for presenting your projects.

• The final submission will contain an archive named Student1FamilyName_Student1Name_ Student2FamilyName_Student2Name_HW1 with:

· the source files of your project (.cpp and .h) and not the object files (.o) or executables (.exe)

· a README file in which you will specify all the functional sections of the project, together with instructions for the user; additionally, if you have parts of the homework that don't work, you may offer solution ideas for a partial score on these sections.

• For all questions regarding the project, communicate by email with your lab assistant.

• Warning: we will use plagiarism detection software on your submissions. Copied homework will be marked with 0 points.

! Observation: The stacks and queues used will be in headers (.h) and will be generic classes (template).

Tasks:

1. Design a stack for double numbers called *AdvancedStack*, which, besides *push, pop, isEmpty, peek* methods, it has methods for returning the minimum element (*min*) and the maximum element (*max*), which should operate in O(1) time, meaning you can't obtain the minimum/maximum by traversing any stack array. Your *AdvancedStack* should be built using the template class Stack from the course. (2p)

2. Finding the path to the examination room at UPB Admission Exam using the Stack data structure (4p)

A building from the UPB campus is represented though a N*M binary matrix of blocks (representing corridors/ stairs, walls and laboratory doors). A student, positioned initially at coordinates (0,0) in the building matrix wants to find the door to his examination room at UPB Admission Exam, which has a numerical code (other than 0 and 1): e.g. 101. In the building matrix, we can have 3 types of values: lab codes, 0 - means corridors/ stairs, 1 - means walls. The student can go through corridors and stairs till the room where one was assigned for the exam. The student can move in any direction (not

diagonally) to any matrix block provided the block is not a dead end (walls and doors different from the one he/she is looking for). The task is to check if any path exists so that the student can reach the examination room and print the path, using the stack data structure.

The input will be 3 building matrices and the code of the student's examination room: the first 2 letters represent the building and the other 3 represent the numerical code inside the building, e.g. CJ101.

Example of building matrix for CJ building:

```
CJbuilding[4][5] = {
        {0, 104, 0, 0, 1},
        {0, 0, 0, 1, 0},
        {1, 0, 1, 101, 102},
        {0, 0, 0, 0, 0}
        }
```

Example of output, when the students searches for CJ101:

Path Found!

The path can be: (0, 0) -> (1, 0) -> (1, 1) -> (2, 1) -> (3, 1) -> (3, 2) -> (3, 3) -> (2, 3)

3. The teachers from UPB want to stock all the available rooms in the campus, which can be used as examination rooms for admission. The room are stocked in a priority queue. Implement a template class called PriorityQueue using the AdvancedStack class: the rooms are added to the queue in the classical way, at the end of the queue, but when deleting them, the room which is the closest to the building entrance door is selected. In order to calculate the distance between a room and the building entrance door, the distance formula between 2 points is used. The building entrance door has coordinates (0,0). The room doors have coordinates equal to the line and the column in the matrix building. (4p)