IIT Ropar
CSL201 Data Structures
Semester 1, AY 2018/19

Lab Assignment 2 - 40 marks Due on September 25th, 11:59 PM

Objective

The objective is to learn to implement priority queue with a heap.

Instructions

1. You are to use C++ programming language to complete the assignment.
2. Provide a Makefile to compile your final code.
3. All function and class declarations should be in ".hpp" files while the definitions should be in ".cpp" files. It is recommended that you have one header file and one cpp file for each class. If the class is templated, declare it in the hpp file itself.
4. This is an individual assignment. You can have high level discussions with other students, though.
5. Include a "Readme.txt" file on how to compile and run the code.
6. Upload your submission to moodle by the due date and time.
7. Late submission is not allowed. No assignment will be accepted through email after submission system closes. I suggest you to upload the assignment at least 1 hour in advance.
8. If you find assignment description is vague, take appropriate assumption and write that in the Readme. Clearly state assumptions and tell those assumptions during viva.
9. If any student asks for deadline extension or sends assignment through email, he or she will get 5% penalty.
10. I should be able to use the class you have implemented in my program the way we use STL classes. There will be a master program to test the classes.

Program Description

In this assignment, you need to design and implement a templated adaptable priority queue using heap. Along with standard priority queue functions, it should also support:
1. remove() function
2. replace() function
3. a special constructor that takes an array of objects as input and uses bottom-up approach to build a heap in O(n).

Your main program should do the following:
1. Accept both lec.txt and lab.txt files as input. It should read all the items, crate objects, store them in an array, and pass to heap for construction.
2. You can see that you need to implement two separate comparators for both type of objects. Compare labs based on the area and lectures based on the number of chairs.
3. The main program interface should also allow the user to add elements dynamically and remove minimum.
4. The program should allow user replace or remove the element just added (you will need the functionality of locator here).
5. There should also be a print function that prints the names (space separated) in level order, i.e., first line prints root, second line prints level 1 node element name, second line level 2, etc. You can see an example in the book how to print a binary tree content for help.

Labs have the following attributes: name, area, number of doors, location (X,Y), and number of computers.
Lecture halls have the following attributes: name, area, number of doors, location (X,Y), number of chairs.

The interface of the modified queue is given in file pQueue.hpp.