January 2023 CSE 208
Offline Assignment on Priority Queue

Deadline: Saturday, 12^{th} August, 2023, 11:55 PM

In this assignment, you have to implement a max priority queue using the binary heap data structure. You need to implement the following functions:

Insert(x): Inserts an element into the priority queue according to the priority of its key.

FindMax(): Returns the element with largest key but does not remove it.

ExtractMax(): Returns the element with largest key and delete it from the heap.

IncreaseKey(i, newKey): Increase the key of the i^{th} element to newKey, and relocate it to maintain heap property.

DecreaseKey(i, newKey): Decrease the key of the i^{th} element to newKey, and relocate it to maintain heap property.

Print(): Print the heap.

Sort(): Sort the elements of the heap in non-dreceasing order in-place, i.e., you are not allowed to use more than $\mathcal{O}(1)$ additional memory.

You have to implement Print in $\mathcal{O}(n)$ time, Sort in $\mathcal{O}(n \log n)$ time and all other operations in $\mathcal{O}(\log n)$ time, where n is the current size of the heap. For increasing and decreasing keys, you will interpret the current position as 1-indexed. That is, the maximum element will be at index 1, its two children will be at indices 2 and 3 and so on.

You are advised to follow C++ object oriented paradigm in your code. Specially, your priority queues should handle any `struct` or `class` instance with overloaded comparison operators. The usage of C++ STL is not allowed in this assignment. Use dynamic memory allocation.

Input Format

Prompt the user for a choice input. Use 1 for Insert, 2 for FindMax, 3 for ExtractMax, 4 for IncreaseKey, 5 for DecreaseKey, 6 for Print, 7 for terminate. Ask user to select an operation until option 7 is selected. Also prompt user for input any value which is required for the corresponding operations, i.e., Insert, IncreaseKey, etc. While terminating your program, sort the elements in-place and print them. Note that the heap property will likely not be maintained after sorting. For the purpose of checking your implementation of the offline assignment, you can write a `main` function assuming the inputs will be integers. If the user asks for an invalid operation, e.g., calls IncreaseKey with a new key smaller than the previous one, print appropriate error message. Please look at the attached files for I/O formatting.

Submission Guideline

1. Create a directory with your 7 digit student id as its name
2. Put the source files only into the directory created in step 1
3. Zip the directory (compress in .zip format; .rar, .7z or any other format is not acceptable)
4. Upload the .zip file on moodle.

For example, if your student id is 2105xxx, create a directory named 2105xxx. Put only your source files (.c, .cpp, .cc, .h, etc.) into 2105xxx. Compress 2105xxx into 2105xxx.zip and upload the 2105xxx.zip on MOODLE.

Marks Distribution

	Task Detail	Marks
1	The Seven Intended Operations	$13 \times 7 = 91$
2	Proper Formatting	5
3	Proper Submission	4

For Queries

If you have any questions related to the assignment please first check the queries thread in MOODLE. You should post your confusion in the thread.

Special Instructions

When writing code, it is essential to ensure readability, reusability, and good structure. This involves using appropriate functions to implement algorithms, giving variables meaningful names, adding suitable comments when necessary, and maintaining proper indentation. You will need to use your offline implementation to solve the onlines. There will be a viva too. So, please understand the concepts before you proceed to code.

Please note that you must rely on your own implementation to solve the assigned tasks. It is strictly prohibited to copy code from any source, including friends, seniors, or the internet. **Any form of plagiarism, regardless of its origin or destination, will result in a deduction of 100% marks for the offline assessment**. Moreover, repeated instances of plagiarism may lead to stricter consequences in accordance with departmental policies.