Objective of this lab:

Heaps

Instructions:

- Make a separate project for each task.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines and labels for all input/output.
- Make sure that there are NO dangling pointers or memory leaks in your program.

Note: "YOUR PROGRAM MUST HAVE A MAIN FUNCTION THAT TEST ALL THE IMPLEMENTED FUNCTIONS"

Question1:

In this task, you are going to implement a class for Max Heap. Each node of this Max Heap will contain the roll number, and CGPA of a student. The heap will be organized on the basis of students' CGPAs i.e. the student having the maximum CGPA will be at the root of the heap. The class definitions will look like:

```
class StudentMaxHeap;
class Student
{
        friend class StudentMaxHeap;
        private:
                int rollNo; // Student's roll number
                double cgpa; // Student's CGPA
};
class StudentMaxHeap
{
private:
        Student* st; // Array of students which will be arranged like a MaxHeap
        int currSize; // Current number of students present in the heap
        int maxSize; // Maximum no. of students that can be present in heap
public:
        StudentMaxHeap (int size); // Constructor
        ~StudentMaxHeap(); // Destructor
        bool isEmpty(); // Checks whether the heap is empty or not
        bool isFull(); // Checks whether the heap is full or not
};
```

First of all, implement the constructor, destructor, isEmpty and isFull functions shown above in the class declaration

Question2:

Implement a public member function of the **StudentMaxHeap** class which inserts the record of a new student (with the given roll number and CGPA) in the Max Heap. The prototype of your function should be:

bool insert (int rollNo, double cgpa);

This function should return true if the record was successfully inserted in the heap and it should return false otherwise. The worst-case time complexity of this function should be $\boldsymbol{O(lg\ n)}$. If two students have the same CGPA then their records should be stored in a way such that at the time

of removal if two (or more) students have the **same highest CGPA** then the student with **smaller roll number** should be removed before the students with larger roll numbers.

Question3:

Now, implement a public member function to remove that student's record from the Max Heap which has the highest CGPA. The prototype of your function should be:

bool removeBestStudent (int& rollNo, double& cgpa);

Before removing the student's record, this function will store the roll number and CGPA of the removed student in its two reference parameters. It should return true if the removal was successful and it should return false otherwise. The worst-case time complexity of this function should also be $\boldsymbol{O(lg\ n)}$.

Question4:

Implement the following two public member functions of the **StudentMaxHeap** class:

void heapify (int i)

This function will convert the subtree rooted at index i into a Max-Heap. This function will assume that the left-subtree and the right-subtree of index i are already valid Max-Heaps.

void buildHeap (Student* st, int n)

This function will take as array of students (st) and its size (n) as parameters. This function should build a Max-Heap containing the records of all n students using the algorithm that you have discussed in lecture

Question5:

Implement the following global function:

void heapSort (Student* st, int n) // must be implemented as a global function

This function will take as array of students (st) and its size (n) as parameters. This function should sort the array of students (st) into increasing order (according to CGPA), using the Heap sort algorithm that you have discussed in lecture.

Also write a driver main function to test the working of the above function.

Just for understanding or revision of this concept:

Heap Sort Algorithm

Heap sort processes the elements by creating the min-heap or max-heap using the elements of the given array. Min-heap or max-heap represents the ordering of array in which the root element represents the minimum or maximum element of the array.

Heap sort basically recursively performs two main operations -

- Build a heap H, using the elements of array.
- Repeatedly delete the root element of the heap formed in 1st phase.
- Before knowing more about the heap sort, let's first see a brief description of Heap.

What is a heap?

A heap is a complete binary tree, and the binary tree is a tree in which the node can have the utmost two children. A complete binary tree is a binary tree in which all the levels except the last level, i.e., leaf node, should be completely filled, and all the nodes should be left-justified.

What is heap sort?

- Heapsort is a popular and efficient sorting algorithm.
- The concept of heap sort is to eliminate the elements one by one from the heap part of the list, and then insert them into the sorted part of the list.
- Heapsort is the in-place sorting algorithm.

Working of Heap sort Algorithm

In heap sort, basically, there are two phases involved in the sorting of elements.
By using the heap sort algorithm, they are as follows -

The first step includes the creation of a heap by adjusting the elements of the array.
After the creation of heap, now remove the root element of the heap repeatedly by shifting it to the end of the array, and then store the heap structure with the remaining elements.

GOOD LUCK