

# Data Structures and Algorithms Lab

## Lab 12

## Marks 05

### Instructions

Work in this lab individually. You can use your books, notes, handouts etc. but you are not allowed to borrow anything from your peer student.

### Marking Criteria

Show your work to the instructor before leaving the lab to get some or full credit.

### What you must do

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 based on students' **CGPAs** i.e., the student having the **maximum CGPA** will be at the **root** of the heap.

The class definitions will look like:

```
/**Holds information about a student, including CGPA and roll number.*/
class Student
{
    friend class StudentMaxHeap;

private:
    double cgpa;           /**Student's CGPA*/
    int rollNo;            /**Student's roll number*/
};

/**Manages an array of students arranged like a Max Heap.*/
class StudentMaxHeap
{
private:
    Student* arr;          /**Array of students which, arranged like a Max Heap*/
    int curSize;           /**Current number of students present in the heap*/
    int maxSize;           /**Maximum number 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*/
};
```

You are required to implement the following member functions of the **StudentMaxHeap** class:

**bool insert (int rollNo, double cgpa)**

This function will insert the record of a new student (with the given roll number and CGPA) in the **Max Heap**. This function should return **true** if the record was successfully inserted in the heap and it should return **false** otherwise, with worst case time complexity of  $O(\lg n)$ .

If two students have the same **CGPA** then their records should be stored in a way such that the student with **smaller** roll number comes before the student with **larger** roll number.

**bool remove (int &rollNo, double &cgpa)**

This function will remove the record from **Max Heap** which has the highest **CGPA**. Before removing the student's record, this function will store the **roll number** and **CGPA** of the removed student in its two arguments. 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  $O(\lg n)$ .

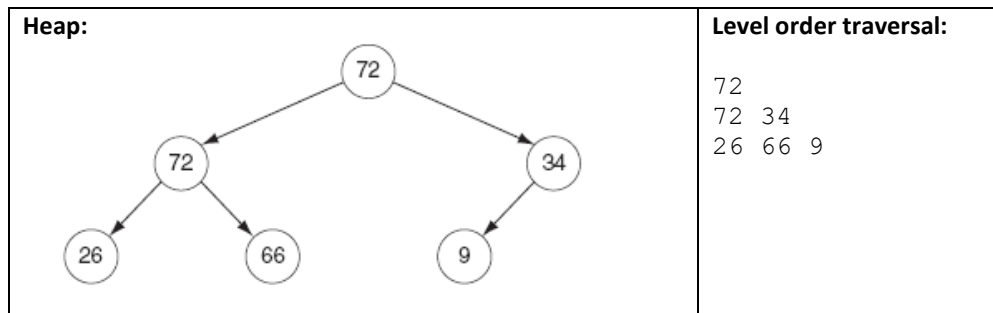
**void displayStudentList()**

This function will display the students list in **descending order** of **CGPAs**. If two students have the same **CGPA**, then the student with **smaller roll number** will be displayed before the student with **larger roll number**.

This function should not affect the **heap** i.e., after the execution of this function the heap should be left in the same way, as it was when this function was called.

```
void levelOrder()
```

This function will perform a **level order** traversal of **StudentMaxHeap** and display the **roll numbers** and **CGPAs** of the students. The output displayed by the level order traversal should follow the format as shown below:



Write a **menu-based** driver function (**menu**) to illustrate the working of different functions of the **StudentMaxHeap** class. The menu should look like:

1. Insert a new student
2. Remove (and display) the student with the Max CGPA
3. Display the list of students (Descending order of CGPA)
4. Display the list of students (Level-order traversal)
5. Exit

Enter your choice:

---

😊😊😊 **BEST OF LUCK** 😊😊😊

---