National University of Computer and Emerging Sciences

Lab Manual

for

Data Structure

Course Instructor	Mr. Razi Uddin
Lab Instructor(s)	Ms. Mamoona Akbar Ms. Mateen Fatima
Section	DS (BSE-3B)
Semester	FALL 2023

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 10

Objectives:

After performing this lab, students shall be able to Practice:

- ✓ Heap ADT

Task 1

Build a min heap class having following structure.

```
class minHeap{
public:

minHeap( ); // default constructor

void buildMinHeap() // It will generate heap from random values stored in the object.

void insertASingleValueInHeap(const int & x); //


bool isEmpty() const; // returns true if it is empty

const T & getMin() const; //returns minimum value this operation should be
performed in O(1)

private:
int* h;
int maxSize, currHeapSize;

// following are the helper functions
void bubble_up(int i); // A recursive method to heapify a subtree with the root at given index. It maintains
heap property during insertion

void bubble_down(int i); // It maintains heap property during deletion

};
```

Now write a main function to check all the functionality of the minHeap.

Task 2

You can use the heap class that was built in above.

TASK 1: Write a function heapsort void heapSort(int arr[], int N);

TASK 2: Given an array, and its size, write a function to sort the array in descending order using heapsort. void heapSortDecending(int arr[], int n);

TASK 3: Write a function to print all smaller values given a number x. Void PrintSmallerThan(int arr[], int n, int x);

Task 3

In this question, you are required to complete the insert, deleteMax, and display functions within the MaxHeap class. You should use a dynamically allocated array (`int* heap`) with a specified capacity to represent the heap and ensure that the max heap property is maintained after each insertion and deletion. The main function is left incomplete, and students need to create an instance of MaxHeap, insert some elements, perform deletions, and display the heap at each step to demonstrate the operations of the max heap.

```
class MaxHeap {
private:
    int* heap;  // Using dynamic array
    int size;   // Current size of the heap
    int capacity; // Maximum capacity of the heap
public:
    // Constructor to initialize the heap with a given capacity
    MaxHeap(int maxCapacity) : size(0), capacity(maxCapacity) {
        heap = new int[capacity];
    }
    // Destructor to free the dynamically allocated memory
```

```
    ~MaxHeap() {
        delete[] heap;
    }
    // Implement the insertion operation for the max heap
    void insert(int value) {
        // TODO: Add the value to the heap and adjust the heap to maintain the max heap property
    }
    // Implement the deletion operation for the max heap
    void deleteMax() {
        // TODO: Remove the maximum element from the heap and adjust the heap accordingly
    }
    // Implement a function to print the elements of the heap
    void display() {
        // TODO: Print the elements of the heap
    }
};
int main() {
    // TODO: Create an instance of MaxHeap with a specified capacity, insert some elements, perform deletions, and display the heap at each step
    return 0;
}
```