NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

PROGRAM: SOFTWARE ENGINEERING

DATA STRUCTURES LAB

LAB TASK-13

SUBMITTED BY:

Name: Ahmed Ali

Roll No: 22P-9318

INSTRUCTOR NAME: Sir Saood Sarwar

A DEPARTMENT OF COMPUTER SCIENCE

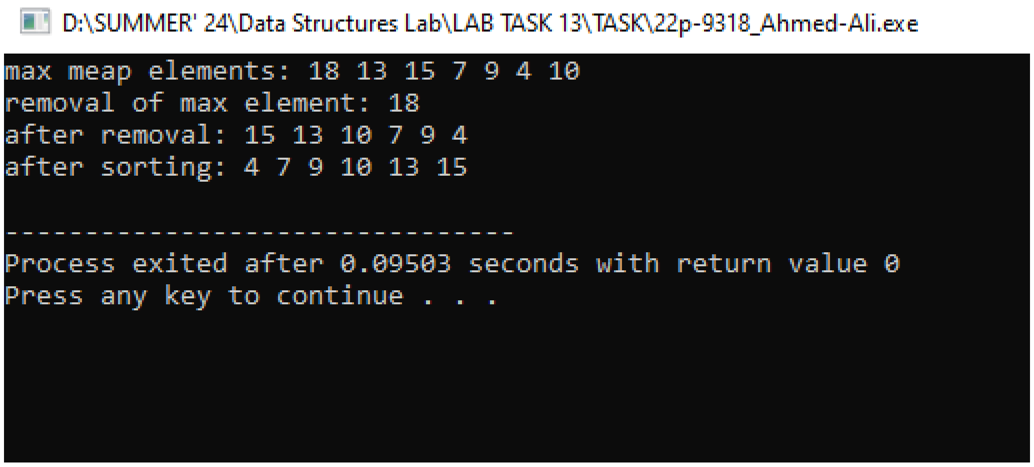
Q1 CODE:

```
//Max Heap: Every parent should be greater or equal to their child, duplication is allowed, At max heap
root is largest

//Min Heap: Every parent should be less than or equal to their child, duplication is allowed, At min heap
root is smallest

// formula to find parent = (i-1)/2

#include<iostream>
using namespace std;

class max_heap
{
        private:
                int *arr;
                int size;
                int n;
                int *root;

        public:
                max_heap(int size)
                {
                        this->size=size;
                        arr= new int[size];
                        n=0;
        root=nullptr;
}

void heapify(int n)
{
        int parent=(n-1)/2;

        if(parent>=0)
        {
                if(arr[n]>arr[parent])
                {
                        swap(arr[n], arr[parent]);
                        heapify(parent);
                }
        }
}

void insert(int data)
{
        if(size==n)
        {
                cout<<"full"<<endl;
                return;
        }
        arr[n]=data;
        n++;
        heapify(n-1);
}
```

```
void display()
{
        for(int i=0; i<n; i++)
        {
                cout<<arr[i]<<" ";
        }
        cout<<endl;
}

void heapify_down(int i)
{
        int greatest=i;
        int left=2*i+1;    //to find left child
        int right=2*i+2; //to find right child

        if(left<n)
        {
                if(arr[left]>arr[greatest])
                {
                        greatest=left;
                }
        }

        if(right<n)
        {
                if(arr[right]>arr[greatest])
                {
                        greatest=right;
                }
                }

                if(greatest!=i)
                {
                        swap(arr[i], arr[greatest]);
                        heapify_down(greatest);
                }
        }

        int remove()
        {
                if(n==0)
                {
                        cout<<"empty"<<endl;
                        return -1;
                }
                if(root==nullptr)
                {
                        root=new int;
                }
                *root=arr[0];
                arr[0]=arr[n-1];
                n--;
                heapify_down(0);
                int val_of_root=*root;
                return val_of_root;
        }

void h_sort()
```

```
                {
                        int actual_size=n;
                        while(n>1)
                        {
                                swap(arr[0], arr[n-1]);
                                n--;
                                heapify_down(0);
                        }
                        n=actual_size;
                }
};
int main()
{
        max_heap h(10);
        h.insert(15);
        h.insert(13);
        h.insert(10);
        h.insert(7);
        h.insert(9);
        h.insert(4);
        h.insert(18);
        cout<<"max meap elements: ";
        h.display();
        int maximum=h.remove();
        cout<<"removal of max element: "<<maximum<<endl;
        cout<<"after removal: ";
        h.display();
        h.h_sort();
        cout<<"after sorting: ";
        h.display();

        return 0;

}
```

Output-01:


D:\SUMMER' 24\Data Structures Lab\LAB TASK 13\TASK\22p-9318_Ahmed-Ali.exe

```
max meap elements: 18 13 15 7 9 4 10
removal of max element: 18
after removal: 15 13 10 7 9 4
after sorting: 4 7 9 10 13 15

--------------------------------
Process exited after 0.09503 seconds with return value 0
Press any key to continue . . .
```