# Mahad Ghauri

233523

DSA Lab Task

#include <iostream>

using namespace std;

template <class T>

class Heap

{

public:

    int arr[100];

    int size;

    Heap()

    {

        arr[0] = -1;

        size = 0;

    }

    void insert(int val)

    {

        int index = ++size;

        arr[index] = val;

        while (index > 1)

        {

            int parent = index / 2;

            if (arr[parent] < arr[index])

            {

                swap(arr[parent], arr[index]);

                index = parent;

            }

            else

            {

                return;

            }

        }

    }

    void deleteHeap()

    {

        if (size == 0)

        {

            cout << "Heap is empty!" << endl;

            return;

        }

        arr[1] = arr[size];

        size--;

        int i = 1;

        while (i < size)

        {

            int leftChild = 2 \* i;

            int rightChild = (2 \* i) + 1;

            int largest = i;

            if (leftChild <= size && arr[largest] < arr[leftChild])

            {

                largest = leftChild;

            }

            if (rightChild <= size && arr[largest] < arr[rightChild])

            {

                largest = rightChild;

            }

            if (largest == i)

            {

                return;

            }

            else

            {

                swap(arr[i], arr[largest]);

                i = largest;

            }

        }

    }

    void print()

    {

        cout << "Max Heap : ";

        for (int i = 1; i <= size; i++)

        {

            cout << arr[i] << " ";

        }

        cout << endl;

    }

};

template <class T>

void findMax(Heap<T> &heap)

{

    if (heap.size == 0)

    {

        cout << "Heap is empty!" << endl;

    }

    else

    {

        cout << "Maximum value in the heap: " << heap.arr[1] << endl;

    }

}

int main()

{

    Heap<int> heap1;

    heap1.insert(80);

    heap1.insert(70);

    heap1.insert(60);

    heap1.insert(25);

    heap1.insert(55);

    heap1.insert(90);

    heap1.insert(10);

    heap1.print();

    findMax(heap1);

    heap1.deleteHeap();

    heap1.print();

    findMax(heap1);

    return 0;

}

Output:

