}

//Write C++ program to create Max Heap Tree

```
#include <iostream>

#include <vector>

using namespace std;

class MaxHeap {

private:

    vector<int> heap;

    // Function to heapify up after insertion

    void heapifyUp(int index) {

        if (index == 0) return; // Base case

        int parent = (index - 1) / 2;

        if (heap[parent] < heap[index]) {

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

            heapifyUp(parent); // Recursive call to ensure max-heap property

        }

    }

  // Function to heapify down after deletion

    void heapifyDown(int index) {

        int left = 2 * index + 1;

        int right = 2 * index + 2;

        int largest = index;


        if (left < heap.size() && heap[left] > heap[largest]) {

            largest = left;

        }
        if (right < heap.size() && heap[right] > heap[largest]) {
            largest = right;
        }
        if (largest != index) {
            swap(heap[index], heap[largest]);
            heapifyDown(largest); // Recursive call to maintain max-heap property
        }
    }

public:
    // Function to insert a new element into the heap
    void insert(int value) {
        heap.push_back(value);
        heapifyUp(heap.size() - 1); // Adjust position to maintain max-heap
    }
```
// Function to remove and return the maximum element (root) from the heap
```
    int extractMax() {
        if (heap.empty()) {
            cout << "Heap is empty!" << endl;
            return -1;
        }
        int maxElement = heap[0];
        heap[0] = heap.back();
        heap.pop_back();
        heapifyDown(0); // Adjust position to maintain max-heap
        return maxElement;
    }

    // Function to display the elements of the heap
    void printHeap() {
        for (int i = 0; i < heap.size(); ++i) {
            cout << heap[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    MaxHeap maxHeap;

    // Insert elements into the max heap
    maxHeap.insert(10);
    maxHeap.insert(20);
    maxHeap.insert(15);
    maxHeap.insert(30);
    maxHeap.insert(40);

    cout << "Max Heap after insertions: ";
    maxHeap.printHeap();
    // Extract maximum elements
    cout << "Extracted max: " << maxHeap.extractMax() << endl;
    cout << "Heap after extraction: ";
    maxHeap.printHeap();

    return 0;
}
```

//output

```
Max Heap after insertions: 40 30 15 10 20
Extracted max: 40
Heap after extraction: 30 20 15 10
```

//Write C++ program to create Min Heap Tree

```
#include <iostream>
#include <vector>
using namespace std;
class MinHeap {
private:
    vector<int> heap;
  // Function to heapify up after insertion
    void heapifyUp(int index) {
        if (index == 0) return; // Base case
        int parent = (index - 1) / 2;
        if (heap[parent] > heap[index]) {
            swap(heap[parent], heap[index]);
            heapifyUp(parent); // Recursive call to maintain min-heap property
        }

    }


    // Function to heapify down after deletion

    void heapifyDown(int index) {

        int left = 2 * index + 1;

        int right = 2 * index + 2;

        int smallest = index;


        if (left < heap.size() && heap[left] < heap[smallest]) {

            smallest = left;

        }

        if (right < heap.size() && heap[right] < heap[smallest]) {

            smallest = right;

        }

        if (smallest != index) {

            swap(heap[index], heap[smallest]);

            heapifyDown(smallest); // Recursive call to maintain min-heap property

        }

    }


public:

    // Function to insert a new element into the heap

    void insert(int value) {

        heap.push_back(value);
        heapifyUp(heap.size() - 1); // Adjust position to maintain min-heap
    }

    // Function to remove and return the minimum element (root) from the heap
    int extractMin() {
        if (heap.empty()) {
            cout << "Heap is empty!" << endl;
            return -1;
        }
        int minElement = heap[0];
        heap[0] = heap.back();
        heap.pop_back();
        heapifyDown(0); // Adjust position to maintain min-heap
        return minElement;
    }

    // Function to display the elements of the heap
    void printHeap() {
        for (int i = 0; i < heap.size(); ++i) {
            cout << heap[i] << " ";
        }
        cout << endl;
    }
};
```

```
int main() {
    MinHeap minHeap;
```

// Insert elements into the min heap

```
    minHeap.insert(20);
    minHeap.insert(15);
    minHeap.insert(30);
    minHeap.insert(40);
    minHeap.insert(10);
   cout << "Min Heap after insertions: ";
    minHeap.printHeap();
```

 // Extract minimum element

```
    cout << "Extracted min: " << minHeap.extractMin() << endl;
    cout << "Heap after extraction: ";
    minHeap.printHeap();
  return 0;
}
```

//output

```
Min Heap after insertions: 10 15 30 40 20
Extracted min: 10
Heap after extraction: 15 20 30 40
```