THIS IS MY CODE:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

// Function to print the heap
void printHeap(const vector<int>& heap) {
   cout << "Current Heap: ";
   for (int val : heap) {
      cout << val << " ";
   }
   cout << endl;
}

// Function to heapify a subtree rooted at index `i` (Max-Heap)
void maxHeapify(vector<int>& heap, int n, int i) {
   int largest = i; // Initialize largest as root
   int left = 2 * i + 1;
   int right = 2 * i + 2;

   // Check if left child exists and is greater than root
   if (left < n && heap[left] > heap[largest])
      largest = left;

   // Check if right child exists and is greater than the current largest
   if (right < n && heap[right] > heap[largest])
      largest = right;
```

```
    // If largest is not root
    if (largest != i) {
        swap(heap[i], heap[largest]);
        maxHeapify(heap, n, largest); // Recursively heapify the affected subtree
    }
}

// Function to insert a new element into a Max-Heap
void insertMaxHeap(vector<int>& heap, int value) {
    heap.push_back(value);
    int i = heap.size() - 1;

    // Fix the heap property if violated
    while (i > 0 && heap[(i - 1) / 2] < heap[i]) {
        swap(heap[i], heap[(i - 1) / 2]);
        i = (i - 1) / 2;
    }

    cout << "Added mission with urgency level " << value << " to the Max-Heap!" << endl;
    printHeap(heap);
}

// Function to delete the root of a Max-Heap
void deleteRoot(vector<int>& heap) {
    if (heap.empty()) {
        cout << "Heap is empty!" << endl;
        return;
    }
```

```
    int n = heap.size();
    cout << "Removing mission with urgency level " << heap[0] << " from the Max-Heap!" << endl;

    // Replace root with the last element and heapify
    heap[0] = heap[n - 1];
    heap.pop_back();
    maxHeapify(heap, heap.size(), 0);
    printHeap(heap);
}

// Function to convert Max-Heap to Min-Heap
void convertToMinHeap(vector<int>& heap) {
    // Invert values and rebuild heap as Max-Heap
    for (int& val : heap)
        val = -val;

    for (int i = heap.size() / 2 - 1; i >= 0; --i)
        maxHeapify(heap, heap.size(), i);

    // Revert values back to positive
    for (int& val : heap)
        val = -val;

    cout << "Converted to Min-Heap: ";
    printHeap(heap);
}

// Main function
```

```
int main() {
    vector<int> heap;

    // Insert elements
    insertMaxHeap(heap, 50);
    insertMaxHeap(heap, 20);
    insertMaxHeap(heap, 70);
    insertMaxHeap(heap, 10);
    insertMaxHeap(heap, 40);

    // Delete the root
    deleteRoot(heap);

    // Convert to Min-Heap
    convertToMinHeap(heap);

    return 0;
}
```