Merge Two Binary Max Heaps

You are given two integer arrays 'ARR1' and 'ARR2' of size 'N' and 'M' respectively, Both 'ARR1' and 'ARR2' represent a Max-Heap. Your task is to merge the two arrays firstly keep all the elements of the 'ARR1' (without changing order) than elements of array 'ARR2' (without changing order), and then find the Max-Heap obtained by merging arrays. Print an array of sizes 'N + M', representing the Max-Heap obtained after merging.
A binary Max-Heap is a complete binary tree in which the value in each internal node is greater than or equal to the values in the children of that node.
Max-Heaps are usually represented by an array, as follows :

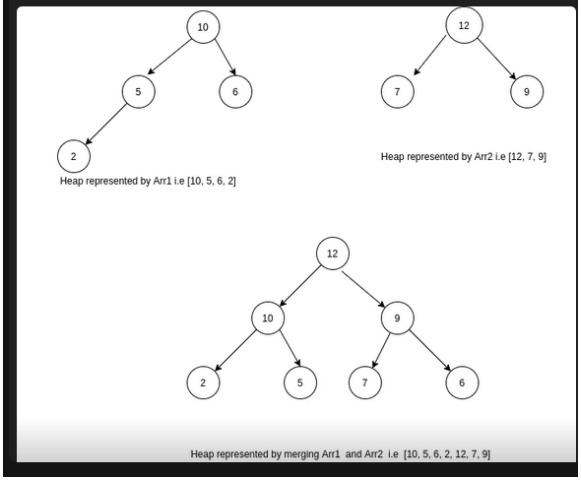
1. Each element in the array represents a node of the heap.

2. Element at index 0 represent the root of the heap.

3. If a node is represented by elements at index 'i' then its left and right child is represented by elements at indices '2*i + 1' and ' 2*i + 2' respectively.

Example :

Consider 'ARR1' = [10, 5, 6, 2] and 'ARR2' = [12, 7, 9].



→ Approach 1:

↳ Use 'priority_queue'

```
vector<int> mergeHeap(int n, int m, vector<int> &arr1, vector<int> &arr2) {
  vector<int> ans;

    priority_queue<int> pq;
    for (int i = 0; i < n; i++) {
      pq.push(arr1[i]);
    }
    for (int i = 0; i < m; i++) {
      pq.push(arr2[i]);
    }

    while(!pq.empty()) {
        int front = pq.top();
        pq.pop();
        ans.push_back(front);
    }

  return ans;
}
```

Partially accepted.

→ Approach 2:-

```
#include <bits/stdc++.h>

void heapify(vector<int> &ans, int index) {
    if(index >= ans.size()) return;
  int largest = index;

  int left = 2 * index + 1;
  int right = 2 * index + 2;

  if (left < ans.size() && ans[left] > ans[largest])
    largest = left;
  if (right < ans.size() && ans[right] > ans[largest])
    largest = right;

  if (largest != index) {
    swap(ans[largest], ans[index]);
    heapify(ans, largest);
  }
}

vector<int> mergeHeap(int n, int m, vector<int> &arr1, vector<int> &arr2) {
  vector<int> ans;
  ans.insert(ans.begin(), arr1.begin(), arr1.end());
  auto it = ans.begin() + n;
  ans.insert(it, arr2.begin(), arr2.end());
  int size = ans.size();

  for (int i = size / 2 - 1; i >= 0; i--) {
    heapify(ans, i);
  }

  return ans;
}
```