#include <iostream>

#include <vector>

using namespace std;

class MaxHeap {

private:

    vector<int> heap;

    // Get the parent index of a node

    int parent(int index) {

        return (index - 1) / 2;

    }

    // Get the left child index of a node

    int leftChild(int index) {

        return 2 \* index + 1;

    }

    // Get the right child index of a node

    int rightChild(int index) {

        return 2 \* index + 2;

    }

    // Heapify up operation (to maintain heap property after insertion)

    void heapifyUp(int index) {

        while (index > 0 && heap[parent(index)] < heap[index]) {

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

            index = parent(index);

        }

    }

    // Heapify down operation (to maintain heap property after deletion)

    void heapifyDown(int index) {

        int maxIndex = index;

        int left = leftChild(index);

        int right = rightChild(index);

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

            maxIndex = left;

        }

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

            maxIndex = right;

        }

        if (index != maxIndex) {

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

            heapifyDown(maxIndex);

        }

    }

public:

    // Insert an element into the max-heap

    void insert(int value) {

        heap.push\_back(value);

        heapifyUp(heap.size() - 1);

    }

    // Delete the element at the specified index from the max-heap

    void removeAt(int index) {

        if (index < 0 || index >= heap.size()) {

            cout << "Invalid index. Cannot remove element." << endl;

            return;

        }

        heap[index] = heap.back();

        heap.pop\_back();

        heapifyDown(index);

        heapifyUp(index);

    }

    // Display the elements of the max-heap

    void display() {

        cout << "Max-Heap: ";

        for (int i = 0; i < heap.size(); ++i) {

            cout << heap[i] << " ";

        }

        cout << endl;

    }

};

int main() {

    MaxHeap maxHeap;

    int numElements;

    cout << "Enter the number of elements to insert into the max-heap: ";

    cin >> numElements;

    cout << "Enter the elements to insert:" << endl;

    for (int i = 0; i < numElements; ++i) {

        int element;

        cin >> element;

        maxHeap.insert(element);

    }

    // Display the max-heap before deletion

    cout << "Max-Heap before deletion: ";

    maxHeap.display();

    // Ask for the index of the element to delete

    int indexToDelete;

    cout << "Enter the index of the element to delete: ";

    cin >> indexToDelete;

    // Delete the element at the specified index from the max-heap

    maxHeap.removeAt(indexToDelete);

    // Display the max-heap after deletion

    cout << "Max-Heap after deletion: ";

    maxHeap.display();

    return 0;

}