AVANISH RAJ SRIVASTAVA
BT22CSH031
ASSIGNMENT 5

1)

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

using namespace std;

void heapify(vector<int>& arr, int n, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] < arr[smallest])
        smallest = left;

    if (right < n && arr[right] < arr[smallest])
        smallest = right;

    if (smallest != i) {
        swap(arr[i], arr[smallest]);
        heapify(arr, n, smallest);
    }
}

void buildHeap(vector<int>& arr) {
    int n = arr.size();

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

void insert(vector<int>& arr, int value) {
    arr.push_back(value);

    int index = arr.size() - 1;
    while (index > 0 && arr[index] < arr[(index - 1) / 2]) {
        swap(arr[index], arr[(index - 1) / 2]);
        index = (index - 1) / 2;
    }
```

```
}

int main() {
    vector<int> arr = {1, 5, 6, 8, 9, 7, 3};


    buildHeap(arr);

    cout << "Min Heap: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;


    int newValue = 4;
    insert(arr, newValue);

    cout << "Min Heap after inserting " << newValue << ": ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Status Successfully executed **Date** 1695746281 **Time** 0.0000 sec **Mem** 3476 kB

Output

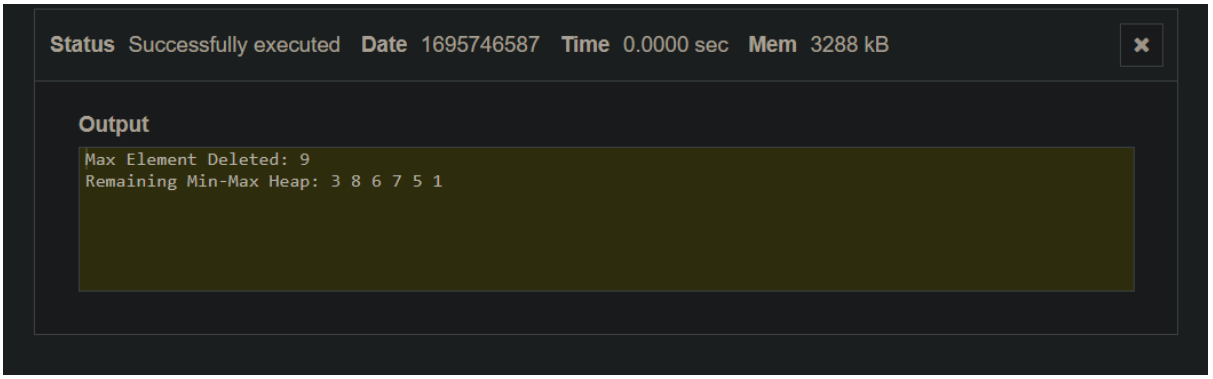
```
Min Heap: 1 5 3 8 9 7 6
Min Heap after inserting 4: 1 4 3 5 9 7 6 8
```

2)

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

using namespace std;

void heapify(vector<int>& arr, int n, int i, bool isMinLevel) {
    int largest = i;
    int left = 2 * i + 1;
```

```
    int right = 2 * i + 2;

    if (isMinLevel) {
        if (left < n && arr[left] < arr[largest])
            largest = left;
        if (right < n && arr[right] < arr[largest])
            largest = right;
    } else {
        if (left < n && arr[left] > arr[largest])
            largest = left;
        if (right < n && arr[right] > arr[largest])
            largest = right;
    }

    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest, !isMinLevel);
    }
}

int deleteMax(vector<int>& arr) {
    if (arr.empty()) {
        cerr << "Heap is empty!" << endl;
        return -1; // Return a sentinel value to indicate an empty heap.
    }

    int maxElement = arr[0];
    int lastIndex = arr.size() - 1;


    swap(arr[0], arr[lastIndex]);


    arr.pop_back();


    bool isMinLevel = true; // Root is at the min-level
    heapify(arr, arr.size(), 0, isMinLevel);

    return maxElement;
}

int main() {
    vector<int> minMaxHeap = {9, 8, 6, 7, 5, 1, 3};

    cout << "Max Element Deleted: " << deleteMax(minMaxHeap) << endl;

    cout << "Remaining Min-Max Heap: ";
```

```
    for (int num : minMaxHeap) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```



3)

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

using namespace std;

void heapify(vector<int>& arr, int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

void buildMaxHeap(vector<int>& arr) {
    int n = arr.size();
```

```
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }
}

void heapSort(vector<int>& arr) {
    int n = arr.size();


    buildMaxHeap(arr);


    for (int i = n - 1; i > 0; i--) {

        swap(arr[0], arr[i]);


        heapify(arr, i, 0);
    }
}

int main() {
    vector<int> arr = {12, 11, 13, 5, 6, 7};

    cout << "Original Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    heapSort(arr);

    cout << "Sorted Array: ";
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Status Successfully executed **Date** 1695746694 **Time** 0.0000 sec **Mem** 3216 kB

Output

```
Original Array: 12 11 13 5 6 7
Sorted Array: 5 6 7 11 12 13
```