Heapsort in C++

```
#include <iostream>
using namespace std;

void heapify(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 heapSort(int arr[], int n) {
    for(int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for(int i = n - 1; i > 0; i--) {
        swap(arr[0], arr[i]);
        heapify(arr, i, 0);
    }
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr)/sizeof(arr[0]);

    heapSort(arr, n);

    cout << "Sorted array is \n";
    for(int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    return 0;
}
```

Step-by-Step Dry Run

✅ Step 1: Build Max Heap

Indices:

0: 12 1: 11 2: 13 3: 5 4: 6 5: 7

Start from i = 2 (last non-leaf node)

i	Heapify Subtree	Max-Heap after heapify
2	[13, 7]	No change
1	[11, 5, 6]	No change
0	[12, 11, 13, 5, 6, 7]	swap 12 with 13 → heapify(2) swaps 12 with 7 → Done

🔺 Max Heap Built:

[13, 11, 7, 5, 6, 12]

✅ Step 2: Extract Elements & Heapify

We now swap root with last element and reduce heap size (n--) after each step:

i	Swap arr[0] & arr[i]	Array after swap	Heapify to max heap
5	swap(13, 12)	[12, 11, 7, 5, 6, 13]	→ heapify → [11, 12, 7...] → [11, 6, 7, 5, 12, 13]
4	swap(11, 6)	[6, 5, 7, 11, 12, 13]	→ heapify → [7, 5, 6...]
3	swap(7, 5)	[5, 6, 7, 11, 12, 13]	→ heapify → [6, 5, ...]
2	swap(6, 5)	[5, 6, 7, 11, 12, 13]	→ heapify → [5, 6, ...] (already heap)
1	swap(5, 5)	Done	

✅ Final Output
Sorted array is
5 6 7 11 12 13

Sorted array is

5 6 7 11 12 13