Задача.

С помощью кучи мы можем быстро находить максимальный элемент в списке. Тогда давайте будем находить максимальный элемент, удалять его из кучи, добавляя в итоговый массив. И повторять этот алгоритм, пока куча не опустеет. Так как мы каждый раз доставали максимальный элемент из кучи, то значит итоговый массив будет отсортированным.

```
#include <iostream>
#include <vector>
#include <iterator>
#include <tuple>
#include <climits>
typedef std::vector<long>::iterator Iter;
class Heap {
public:
    Heap(std::vector<long> & arr) : array(arr) {
        heap_begin = arr.begin();
        heap_end = arr.end();
        build_heap();
    }
    static void sort(std::vector<long> & arr) {
        Heap h(arr);
        for(Iter i = h.heap_end - 1; i != h.heap_begin; i--) {
            std::swap(*h.heap_begin, *i);
            h.heap_end--;
            h.heapify(h.heap_begin);
        }
    }
private:
    Iter left(Iter i) {
        Iter l = heap_begin + (i - heap_begin) * 2 + 1;
        return l >= heap_end ? heap_end : l;
    }
    Iter rigth(Iter i) {
        Iter r = heap_begin + (i - heap_begin) * 2 + 2;
        return r >= heap_end ? heap_end : r;
    }
    Iter parent(Iter i) {
        return heap_begin + (i - heap_begin - 1) / 2;
    }
    void heapify(Iter e) {
        Iter l = left(e), r = rigth(e);
        Iter largest;
        if(l != heap_end and *l > *e) largest = l;
        else largest = e;
        if(r != heap_end and *r > *largest) largest = r;
        if(largest != e) {
            std::swap(*e, *largest);
            heapify(largest);
        }
    }
    void build_heap() {
        heap_end = array.end();
        for(Iter i = heap_begin + (heap_end-heap_begin)/2 + 1; i != heap_begin -
1; i--) {
            heapify(i);
        }
    }
    std::vector<long>& array;
    Iter heap_begin;
```

```
    Iter heap_end;
};
int main() {
    std::vector<long> array{4, 2, 7, 15, 78, 96, 32, 15, 32, 48, 15, 52, 74, 61,
8, 4, 9, 8, 4, 2};
    Heap::sort(array);
    for(auto i : array) {
        std::cout << i << ' ';
    }
    std::cout << std::endl;
    return 0;
}
```