#include <iostream>

#include<algorithm>

#include<vector>

using namespace std;

void downheapifyHelper(vector<int> &heap,int idx,int heapSize){

int leftidx=2\*idx+1;

int rightidx=2\*idx+2;

if(leftidx>=heapSize && rightidx>=heapSize){

return;

}

int largestidx=idx;

if(leftidx<heapSize && heap[leftidx]>heap[largestidx]){

largestidx=leftidx;

}

if(rightidx<heapSize && heap[rightidx]>heap[largestidx]){

largestidx=rightidx;

}

if(largestidx==idx)

return;

swap(heap[largestidx],heap[idx]);

downheapifyHelper(heap,largestidx,heapSize);

}

void downheapify(vector<int> &heap,int idx){

int n=heap.size();

int leftidx=2\*idx+1;

int rightidx=2\*idx+2;

if(leftidx>=n && rightidx>=n){

return;

}

int largestidx=idx;

if(leftidx<n && heap[leftidx]>heap[largestidx]){

largestidx=leftidx;

}

if(rightidx<n && heap[rightidx]>heap[largestidx]){

largestidx=rightidx;

}

if(largestidx==idx)

return;

swap(heap[largestidx],heap[idx]);

downheapify(heap,largestidx);

}

//time complexity of building heap in such manner is O(n)

void buildheapoptimise(vector<int> &heap){

for(int i=heap.size()-1;i>=0;i--){

downheapify(heap,i);

}

}

void heapSort(vector<int> &heap){

int heapSize=heap.size();

buildheapoptimise(heap);

for(int i=heap.size()-1;i>=0;i--){

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

heapSize--;

downheapifyHelper(heap,0,heapSize);

}

}

void display(vector<int> &heap){

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

cout<<heap[i]<<" ";

}

cout<<endl;

}

int main() {

vector<int> heap;

int n,x;

cin>>n;

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

cin>>x;

heap.push\_back(x);

}

display(heap);

heapSort(heap);

display(heap);

return 0;

}

Input-

11

100 40 43 5 2 9 20 10 20 35 38

Output-

100 40 43 20 38 9 20 10 5 35 2

2 5 9 10 20 20 35 38 40 43 100