```cpp
//日期：2018/ 时间：
/*
向下调整：保持根节点始终大于两个叶节点。一直往下。
向上调整：如果叶节点大于父亲节点，就交换位置。使得父节点始终大于子节点

建堆：      从n/2处开始向下调整。调整完成后，最大的元素就在根节点
插入：      把节点放在堆的最后，从这个最后的位置开始 向上调整
删除堆顶元素：      把第一个元素位置删除，把最后一个节点放到第一个位置，然后从此开始 向下调整

堆排序：    按照“删除”的方法，删除首元素，并把删除后的元素放到最后。从第一个元素开始向下调整。

*/
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;

const int maxn = 100;
int heap[maxn],n=10;        //heap为堆，n为元素个数

//对heap数组在[low,high]范围进行向下调整
//其中low为欲调整节点 数组下标,high为堆的最后一个节点的数组下标
void downAdjust(int low,int high){
    int i=low,j = i*2;  //i为欲调整节点，j为其左孩子。下面j存放i的左右孩子中较大孩子的坐标‘
    while(j<=high){     //存在孩子节点
        //如果右孩子存在，且右孩子节点的值大于左孩子
        if(j+1 <= high && heap[j+1]>heap[j]){
            j=j+1;
        }

        //如果孩子中最大权值比欲调整节点i大
        if(heap[j] > heap[i]){
            swap(heap[i],heap[j]);  //交换两个节点
            i=j;        //从上往下，继续调整
            j=2*i;
        } else {
            break;      //孩子节点的权值都比欲调整节点i小，调整结束
        }
    }
}

//建堆
void createHeap(){
    for(int i=n/2;i>=1;i--){
        downAdjust(i,n);
    }
}

//删除堆顶元素,把最后一个元素放到堆顶，然后向下调整
void deleteTop(){
    heap[1] = heap[n--];
    downAdjust(1,n);
}
```

```
54
55  //插入元素，把元素插入到堆的最后，然后向上调整
56  //其中low一般设为1，high表示欲调整节点的数组下标
57  void upAdjust(int low,int high){
58      int i=high,j=i/2;           //i为欲调整节点，j为其父亲
59      while(j>=low){              //父亲在[low,high]范围内
60          //父亲权值小于欲调整节点i的权值
61          if(heap[j]<heap[i]){
62              swap(heap[j],heap[i]);
63              i=j;
64              j=i/2;
65          } else {
66              break;
67          }
68      }
69  }
70
71  void insert(int x){
72      heap[++n] = x;
73      upAdjust(1,n);
74  }
75
76  //堆排序思想：建堆->输出堆顶元素->把堆底元素放到堆顶->从上往下调整->输出堆顶元素->.....
77  void heapSort(){
78      createHeap();
79      for(int i=n;i>=1;i--){
80          swap(heap[i],heap[1]);
81          downAdjust(1,i-1);
82      }
83  }
84
85  int main(){
86      scanf("%d",&n);
87      for(int i=1;i<=n;i++){
88          scanf("%d",&heap[i]);
89      }
90      heapSort();
91      for(int i=1;i<=n;i++){
92          printf("%d",heap[i]);
93          if(i!=n)
94              printf(" ");
95          else printf("\n");
96      }
97
98      return 0;
99  }
100
101
```