//快速排序

#include <iostream>

using namespace std;

void quick\_sort(int s[], int l, int r)

{

if (l < r)

{

int i = l, j = r, x = s[l];

while (i < j)

{

while (i < j && s[j] >= x) // 从右向左找第一个小于x的数

j--;

if (i < j)

s[i++] = s[j];

while (i < j && s[i] < x) // 从左向右找第一个大于等于x的数

i++;

if (i < j)

s[j--] = s[i];

}

s[i] = x;

quick\_sort(s, l, i - 1); // 递归调用

quick\_sort(s, i + 1, r);

}

}

int main()

{

int a[5] = { 1, 5, 9, 2, 10 };

quick\_sort(a, 0, 4);

for (int i = 0; i < 5; i++)

cout << a[i] << " ";

}

//堆排序

#include <iostream>

#include<algorithm>

using namespace std;

void HeapAdjust(int \*a, int i, int size) //调整堆

{

int lchild = 2 \* i; //i的左孩子节点序号

int rchild = 2 \* i + 1; //i的右孩子节点序号

int max = i; //临时变量

if (i <= size / 2) //如果i是叶节点就不用进行调整

{

if (lchild <= size&&a[lchild]>a[max])

{

max = lchild;

}

if (rchild <= size&&a[rchild]>a[max])

{

max = rchild;

}

if (max != i)

{

swap(a[i], a[max]);

HeapAdjust(a, max, size); //避免调整之后以max为父节点的子树不是堆

}

}

}

void BuildHeap(int \*a, int size) //建立堆

{

int i;

for (i = size / 2; i >= 1; i--) //非叶节点最大序号值为size/2

{

HeapAdjust(a, i, size);

}

}

void HeapSort(int \*a, int size) //堆排序

{

int i;

BuildHeap(a, size);

for (i = size; i >= 1; i--)

{

//cout<<a[1]<<" ";

swap(a[1], a[i]); //交换堆顶和最后一个元素，即每次将剩余元素中的最大者放到最后面

//BuildHeap(a,i-1); //将余下元素重新建立为大顶堆

HeapAdjust(a, 1, i - 1); //重新调整堆顶节点成为大顶堆

}

}

int main()

{

//int a[]={0,16,20,3,11,17,8};

//0 16 20 3 11 17 8

int a[100];

int size;

cin >> size;

for (int i = 1; i <= size; i++)

cin >> a[i];

HeapSort(a, size);

for (int i = 1; i <= size; i++)

cout << a[i] << " ";

cout << endl;

//while (scanf("%d", &size) == 1 && size>0)

//{

// int i;

// for (i = 1; i <= size; i++)

// cin >> a[i];

// HeapSort(a, size);

// for (i = 1; i <= size; i++)

// cout << a[i] << " ";

// cout << endl;

//}

return 0;

}