//反转链表

/\*

struct ListNode {

int val;

struct ListNode \*next;

ListNode(int x) :

val(x), next(NULL) {

}

};\*/

class Solution {

public:

ListNode\* ReverseList(ListNode\* pHead) {

if(pHead == nullptr)

return pHead;

ListNode\* pbefore = nullptr;

ListNode\* pnode = pHead;

ListNode\* pnewhead = nullptr;

while(pnode != nullptr)

{

ListNode\* pnext = pnode->next;

if(pnext==nullptr) pnewhead = pnode;

pnode->next = pbefore;

pbefore = pnode;

pnode = pnext;

}

return pnewhead;

}

};

//归并排序

#include <iostream>

using namespace std;

// 合并数组，排好序，然后在拷贝到原来的数组array

void MergeArray(int array[], int start, int end, int mid, int temp[]) {

int i = start;

int j = mid + 1;

int k = 0;

while (i <= mid && j <= end) {

if (array[i] < array[j]) {

temp[k++] = array[i++];

}

else {

temp[k++] = array[j++];

}

}

while (i <= mid) {

temp[k++] = array[i++];

}

while (j <= end) {

temp[k++] = array[j++];

}

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

array[start + i] = temp[i];

}

}

// 归并排序，将数组前半部分后半部分分成最小单元，然后在合并

void MergeSort(int array[], int start, int end, int temp[]) {

if (start < end) {

int mid = (start + end) / 2;

MergeSort(array, start, mid, temp);

MergeSort(array, mid + 1, end, temp);

MergeArray(array, start, end, mid, temp);

}

}

// 在这里创建临时数组，节省内存开销，因为以后的temp都是在递归李使用的。

void MergeSort(int array[], int len) {

int start = 0;

int end = len - 1;

int \*temp = new int[len];

MergeSort(array, start, end, temp);

}

void PrintArray(int array[], int len) {

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

cout << array[i] << " ";

cout << endl;

}

int main() {

int array[] = { 3,5,3,6,7,3,7,8,1 };

MergeSort(array, 9);

PrintArray(array, 9);

return 0;

}