#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<cmath>

#include<cstring>

#include<algorithm>

using namespace std;

ios::sync\_with\_stdio(0);//cincout加速

bool realize(int\*a, int&test)

{

}//下面那个二分的判断函数

int binary\_search(int\* a, int low, int high) {//二分查找，传进来的a是个数组

while (high > low) {

int mid = low + (high - low) / 2;//不容易溢出，且已经处理了边界

if (realize(a, mid))

high = mid;

else low = mid + 1;

}

return high;

}

//泡泡排序 - 稳定

void bubble(int\* p, int n) {//从小到大排序的bubble sort

for (int i = n - 1; i >= 0; i--) {

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

if (p[j] > p[j + 1]) {

int tmp = p[j];

p[j] = p[j + 1];

p[j + 1] = tmp;

}

}

}

}

//快速排序 - 不稳定，这个swap和下面的quick\_sort打包

void swap(int& p, int& q) {

int tmp = p;

p = q;

q = tmp;

}

//快排

void quick\_sort(int\* p, int n) {

if (n <= 1)return;

int t = p[n - 1], l = 0, r = n - 2;

do {

for (; r >= l; l++)if (p[l] > t)break;

for (; r >= l; r--)if (p[r] < t)break;

if (r >= l)swap(p[l], p[r]);

} while (r >= l);

swap(p[l], p[n - 1]);

quick\_sort(&p[0], l);

quick\_sort(&p[l + 1], n - l - 1);

}

//向上取整

int cm(int a, int b) {//向上取整

if (a % b == 0)return a / b;

else return a / b + 1;

}//向上取整函数//向上取整

//归并排序

void merge\_sort(int\* p, int n) {//归并

if (n <= 1) return;//初始化条件

int l = n / 2, r = n - n / 2;//把数组分成两部分

merge\_sort(p, l);//排前半段（前l个）；

merge\_sort(p + l, r);//排后半段；（后n-l个）

int\* pn = new int[n];//把两个合在一起的部分。开一个新数组用于储存

int i = 0, j = l, k = 0;

while (i < l && j < n) {//分别从前面那段的第一个和后面那段的第一个（最小的那个）开始看。

if (p[i] > p[j]) pn[k++] = p[j++];//分别比较前、后两段中选中的那个。找到那个比较小的，依次放在新数组里头

else pn[k++] = p[i++];

}//如果前或后其中一个里面没有数了，把剩下的数依次放到新数组最后

for (; i < l; i++) pn[k++] = p[i];

for (; j < n; j++) pn[k++] = p[j];

for (int x = 0; x < n; x++) p[x] = pn[x];//新数组的值再赋回来。

delete[]pn;

}

//得最大值

int getmax(int a, int b) //得到两个数的最大值

{

return (a) > (b) ? (a) : (b);

}