**1610300c02 夏鹏程**

**总分: 135 / 135**

编程题总分: 135 / 135

7-1答案正确 得分: 10 / 10

利用二分查找找出所给出的数在数组中的下标

**输入格式:**

第一行输入n和m表示数组有n个数据，m表示要对m个数进行查找

**输出格式:**

所有输出在一行完成，行末没有多余空格和多余回车。

**输入样例:**

5 5

1 2 3 4 5

1 2 3 4 5

**输出样例:**

0 1 2 3 4

#include<cstdio>

#include<iostream>

#include<algorithm>

using namespace std;

int main()

{ int n, m, res;

scanf("%d %d", &n, &m);

int a[n];

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

{ scanf("%d", &a[i]);

}

sort(a, a + n);

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

{ int low = 0;

int high = n - 1;

int mid;

scanf("%d", &res);

while (low <= high)

{ mid = (low + high) / 2;

if (a[mid] == res)

{ if (i == m - 1)

printf("%d\n", mid);

else

printf("%d ", mid);

break;

}

else if (a[mid] > res)

{ high = mid - 1;

}

else if (a[mid] < res)

{ low = mid + 1;

}

}

}

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 15 ms | 468KB |
| 1 | 答案正确 | 16 ms | 496KB |
| 2 | 答案正确 | 15 ms | 512KB |
| 3 | 答案正确 | 16 ms | 532KB |
| 4 | 答案正确 | 16 ms | 528KB |

7-2答案正确 得分: 10 / 10

n个元素的数组的最大元素可以用递归计算出来。 定义方法：int max(int x, int y) 它返回x和y两个整数中的较大值。 试用递归编写方法：int arraymax(int[] a, int n) 它使用递归返回数组a的最大元素值。 终止条件：n==2 递归步骤：arraymax=max(max(a[0],...,a[n-2]), a[n-1])

**输入格式:**

第一行的第一个元素是输入元素个数n (1<n<=30)，第二个元素之后是输入n个元素；

**输出格式:**

按格式要求输出相邻两个元素的最大值，例如输出的第一项是a[0]和a[1]之间的最大值；第二项为之前的最大值与a[2]之间的最大值，依次类推，直到最后输出n个元素数组的最大元素值。

**输入样例:**

5 1 3 2 5 3

**输出样例:**

max(1,3)=3 max(3,2)=3 max(3,5)=5 max(5,3)=5 5

#include <cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

#include<iostream>

using namespace std;

int main()

{ int n;

scanf("%d",&n);

int a[40];

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

cin>>a[i];

int flag = a[1];

for( int i = 2 ; i <= n ; i++ )

{ int t = max(flag,a[i]);

printf("max(%d,%d)=%d",flag,a[i],t);

flag = t;

if( i == n )

printf(" %d\n",flag);

else printf(" ");

}

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 184KB |
| 1 | 答案正确 | 2 ms | 384KB |
| 2 | 答案正确 | 2 ms | 188KB |

7-3答案正确 得分: 10 / 10

使用辗转相除法和递归求两个正整数m和n的最大公约数。

**输入格式:**

输入两个正整数m,n。

**输出格式:**

按要求输出辗转相除过程及最终结果，输出结果之间空格分隔。

**输入样例:**

21 35

**输出样例:**

gcd(21,35) gcd(35,21) gcd(21,14) gcd(14,7) 7

import java.util.Scanner;

public class Main {

public static int gcd(int m,int n) {

if(m>n) { int i = n;

if(m%n!=0) {

n = m%n;

m = i;

System.out.print("gcd("+m+","+n+") ");

return gcd(m,n);

}

if(m%n==0) { return n; }

}else {

System.out.print("gcd("+n+","+m+") ");

int i = m;

if(n%m!=0) { n = n%m;

m = i;

System.out.print("gcd("+m+","+n+") ");

return gcd(m,n);

}

}

return 0; }

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int m = input.nextInt();

int n = input.nextInt();

System.out.print("gcd("+m+","+n+") ");

int sum = gcd(m,n);

System.out.println(sum);

} }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 103 ms | 14292KB |
| 1 | 答案正确 | 104 ms | 13540KB |
| 2 | 答案正确 | 103 ms | 14216KB |

7-4答案正确 得分: 10 / 10

设计一个平均时间为O(n)的算法，在n(1<=n<=1000)个无序的整数中找出第k小的数。

提示：函数int partition(int a[],int left,int right)的功能是根据a[left]~a[right]中的某个元素x（如a[left])对a[left]~a[right]进行划分，划分后的x所在位置的左段全小于等于x,右段全大于等于x,同时利用x所在的位置还可以计算出x是这批数据按升非降序排列的第几个数。因此可以编制int find(int a[],int left,int right,int k)函数，通过调用partition函数获得划分点，判断划分点是否第k小，若不是，递归调用find函数继续在左段或右段查找。

**输入格式:**

输入有两行：

第一行是n和k，0<k<=n<=10000

第二行是n个整数

**输出格式:**

输出第k小的数

**输入样例:**

在这里给出一组输入。例如：

10 4

2 8 9 0 1 3 6 7 8 2

**输出样例:**

在这里给出相应的输出。例如：

2

#include <stdio.h>

void swap(int a[], int i, int j)

{ // 交换数组a中元素i与元素j的值

int tmp;

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

int partition(int a[], int lo, int hi)

{ int i = lo-1, pivot = a[lo];

swap(a, lo, hi);

while(lo<hi)

{ if(a[lo] < pivot)

{ i ++;

swap(a, i, lo);

}

lo ++;

}

i ++;

swap(a, i, hi);

return i;

}

int find(int a[], int left, int right, int k)

{ int index;

index = partition(a, left, right);

if(index == k)

return a[index];

else if(index < k)

return find(a, index+1, right, k);

else

return find(a, left, right-1, k);

}

int main()

{ int a[10000];

int n, k, i;

scanf("%d", &n);

scanf("%d", &k);

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

scanf("%d", &a[i]);

printf("%d\n", find(a, 0, n-1, k-1)); }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 128KB |
| 1 | 答案正确 | 2 ms | 128KB |

7-5答案正确 得分: 12 / 12

求整数数组中相差最小的两个元素（称为最接近数）的差值。

**输入格式:**

首先输入数组的长度n，2<=n<=10000，然后输入n个整数作为数组元素。

**输出格式:**

输出数组中相差最小的两个元素的差的绝对值。

**输入样例:**

5

13 26 98 47 55

**输出样例:**

8

#include<stdio.h>

int main()

{ int a[6];

int i,j,y;

int x=100000;

for(i=0;i<6;i++)

scanf("%d",&a[i]);

for(i=0;i<6;i++)

{ for(j=i+1;j<6;j++)

{ if(a[i]<=a[j])

y=a[j]-a[i];

else

y=a[i]-a[j];

if(x>y)

x=y;

}

}

printf("%d",x);

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 1 ms | 128KB |
| 1 | 答案正确 | 1 ms | 128KB |
| 2 | 答案正确 | 1 ms | 128KB |
| 3 | 答案正确 | 1 ms | 128KB |

7-6答案正确 得分: 13 / 13

所谓“螺旋方阵”，是指对任意给定的*N*，将1到*N*×*N*的数字从左上角第1个格子开始，按顺时针螺旋方向顺序填入*N*×*N*的方阵里。本题要求构造这样的螺旋方阵。

**输入格式：**

输入在一行中给出一个正整数*N*（<10）。

**输出格式：**

输出*N*×*N*的螺旋方阵。每行*N*个数字，每个数字占3位。

**输入样例：**

5

**输出样例：**

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

#include<stdio.h>

#define SIZE 10

int main()

{ int a[SIZE][SIZE]={0};

int N;

scanf("%d",&N);

int dx[]={0,1,0,-1};

int dy[]={1,0,-1,0};

int x=0,y=0,nx,ny,i,k=0;

for(i=0;i<N\*N;i++)

{ a[x][y]=i+1;

nx=x+dx[k];

ny=y+dy[k];

if(nx<0 || nx==N || ny<0 || ny==N || a[nx][ny]!=0)

{ k=(k+1)%4;

nx=x+dx[k];

ny=y+dy[k];

}

x=nx;

y=ny;

}

int j;

for(i=0;i<N;i++)

{ for(j=0;j<N;j++)

{ printf("%3d",a[i][j]);

}

printf("\n");

}

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 144KB |
| 1 | 答案正确 | 1 ms | 128KB |
| 2 | 答案正确 | 2 ms | 256KB |
| 3 | 答案正确 | 1 ms | 152KB |

7-7答案正确 得分: 14 / 14

给定*K*个整数组成的序列{ *N*​1​​, *N*​2​​, ..., *N*​*K*​​ }，“连续子列”被定义为{ *N*​*i*​​, *N*​*i*+1​​, ..., *N*​*j*​​ }，其中 1≤*i*≤*j*≤*K*。“最大子列和”则被定义为所有连续子列元素的和中最大者。例如给定序列{ -2, 11, -4, 13, -5, -2 }，其连续子列{ 11, -4, 13 }有最大的和20。现要求你编写程序，计算给定整数序列的最大子列和。

本题旨在测试各种不同的算法在各种数据情况下的表现。各组测试数据特点如下：

* 数据1：与样例等价，测试基本正确性；
* 数据2：102个随机整数；
* 数据3：103个随机整数；
* 数据4：104个随机整数；
* 数据5：105个随机整数；

**输入格式:**

输入第1行给出正整数*K* (≤100000)；第2行给出*K*个整数，其间以空格分隔。

**输出格式:**

在一行中输出最大子列和。如果序列中所有整数皆为负数，则输出0。

**输入样例:**

6

-2 11 -4 13 -5 -2

**输出样例:**

20

#include <stdio.h>

#include <string.h>

#include <algorithm>

#include <math.h>

#include <stdlib.h>

using namespace std;

int a[100200];

int main()

{ int n,m,i,j;

scanf("%d",&n);

int max = 0;

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

{ scanf("%d",&a[i]);

if(max < a[i])

max = a[i];

}

int sum;

for(i = 0;i <= n-2; i++)

{ sum = 0;

for(j = i;j <= n-1;j++)

{

sum = sum+a[j];

if(sum > max)

max = sum;

}

}

printf("%d\n",max);

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 1 ms | 140KB |
| 1 | 答案正确 | 1 ms | 128KB |
| 2 | 答案正确 | 2 ms | 256KB |
| 3 | 答案正确 | 61 ms | 248KB |
| 4 | 答案正确 | 5858 ms | 596KB |

7-8答案正确 得分: 16 / 16

输出整数1至n的字典序全排列。

**输入格式:**

输入整数 n，0<n<10 。

**输出格式:**

输出所有1至n的不重复的全排列，每一行输出一种排列情况，所有的排列情况按字典序输出。最后一行输出全排列的总数。

**输入样例:**

3

**输出样例:**

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

6

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int a[10],p[10],vis[10];//标记为1则说明已固定

int n;

void digui(int l){

int i;

if(l>n){

for(i=1;i<n;i++){

printf("%d ",p[i]);

}

printf("%d\n",p[n]);

}

for(i=1;i<=n;i++){

if(!vis[a[i]]){

p[l]=a[i];

vis[a[i]]=1;

digui(l+1);

vis[a[i]]=0;

}

}

}

int main()

{ int i,j;

scanf("%d",&n);

for(i=1;i<=n;i++){

a[i]=i;

}

memset(vis,0,sizeof(vis));

digui(1);

for(j=n-1;j>0;j--)

{

n=n\*j;

}

printf("%d\n",n);

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 1 ms | 128KB |
| 1 | 答案正确 | 1 ms | 152KB |
| 2 | 答案正确 | 1 ms | 152KB |
| 3 | 答案正确 | 1 ms | 156KB |

7-9答案正确 得分: 20 / 20

社交网络中我们给每个人定义了一个“活跃度”，现希望根据这个指标把人群分为两大类，即外向型（outgoing，即活跃度高的）和内向型（introverted，即活跃度低的）。要求两类人群的规模尽可能接近，而他们的总活跃度差距尽可能拉开。

**输入格式：**

输入第一行给出一个正整数*N*（2≤*N*≤10​5​​）。随后一行给出*N*个正整数，分别是每个人的活跃度，其间以空格分隔。题目保证这些数字以及它们的和都不会超过2​31​​。

**输出格式：**

按下列格式输出：

Outgoing #: N1

Introverted #: N2

Diff = N3

其中N1是外向型人的个数；N2是内向型人的个数；N3是两群人总活跃度之差的绝对值。

**输入样例1：**

10

23 8 10 99 46 2333 46 1 666 555

**输出样例1：**

Outgoing #: 5

Introverted #: 5

Diff = 3611

**输入样例2：**

13

110 79 218 69 3721 100 29 135 2 6 13 5188 85

**输出样例2：**

Outgoing #: 7

Introverted #: 6

Diff = 9359

#include<bits/stdc++.h>

using namespace std;

int a[100002];

int main()

{ int n,m;

scanf("%d",&n);

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

scanf("%d",&a[i]);

sort(a,a+n);

int s1=0,s2=0;

for(int i=0; i<n/2; i++)

s1+=a[i];

for(int i=n/2; i<n; i++)

s2+=a[i];

printf("Outgoing #: %d\n",(n+1)/2);

printf("Introverted #: %d\n",n/2);

printf("Diff = %d\n",s2-s1);

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 244KB |
| 1 | 答案正确 | 2 ms | 256KB |
| 2 | 答案正确 | 29 ms | 640KB |
| 3 | 答案正确 | 23 ms | 596KB |
| 4 | 答案正确 | 2 ms | 240KB |

7-10答案正确 得分: 20 / 20

计算机程序设计能力考试（Programming Ability Test，简称PAT）旨在通过统一组织的在线考试及自动评测方法客观地评判考生的算法设计与程序设计实现能力，科学的评价计算机程序设计人才，为企业选拔人才提供参考标准（网址<http://www.patest.cn）。>

每次考试会在若干个不同的考点同时举行，每个考点用局域网，产生本考点的成绩。考试结束后，各个考点的成绩将即刻汇总成一张总的排名表。

现在就请你写一个程序自动归并各个考点的成绩并生成总排名表。

**输入格式:**

输入的第一行给出一个正整数N（≤100），代表考点总数。随后给出N个考点的成绩，格式为：首先一行给出正整数K（≤300），代表该考点的考生总数；随后K行，每行给出1个考生的信息，包括考号（由13位整数字组成）和得分（为[0,100]区间内的整数），中间用空格分隔。

**输出格式:**

首先在第一行里输出考生总数。随后输出汇总的排名表，每个考生的信息占一行，顺序为：考号、最终排名、考点编号、在该考点的排名。其中考点按输入给出的顺序从1到N编号。考生的输出须按最终排名的非递减顺序输出，获得相同分数的考生应有相同名次，并按考号的递增顺序输出。

**输入样例:**

2

5

1234567890001 95

1234567890005 100

1234567890003 95

1234567890002 77

1234567890004 85

4

1234567890013 65

1234567890011 25

1234567890014 100

1234567890012 85

**输出样例:**

9

1234567890005 1 1 1

1234567890014 1 2 1

1234567890001 3 1 2

1234567890003 3 1 2

1234567890004 5 1 4

1234567890012 5 2 2

1234567890002 7 1 5

1234567890013 8 2 3

1234567890011 9 2 4

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <iostream>

using namespace std;

const int maxn = 41700;

struct ss

{ char num[17];

int s;

int rank;//在当前考点的排名

int mark;//在几号考点

int flag;//总排名

}stu[maxn];

bool cmp1(ss a, ss b)

{ if(a.s == b.s)

{

return strcmp(a.num, b.num) < 0;

}

return a.s > b.s;

}

int main()

{ int N;

int k[117];

int sum = 0;

scanf("%d",&N);

{ int i, j;

int l = 0, p = 0;

for(i = 1; i <= N; i++)

{ scanf("%d",&k[i]);

sum += k[i];

int tt = l;

for(j = 1; j <= k[i]; j++)

{ scanf("%s%d",stu[l].num,&stu[l].s);

stu[l].mark = i;

l++;

}

sort(stu+tt,stu+l,cmp1);

p = 2;

stu[tt].rank = 1;

for(int h = tt+1; h < l; h++)

{ if(stu[h-1].s == stu[h].s)

{ stu[h].rank = stu[h-1].rank;

}

else

{ stu[h].rank = p;

}

p++;

}

}

sort(stu,stu+l,cmp1);

int f = 1;

int flag = 0;

printf("%d\n",sum);

stu[0].flag = 1;

printf("%s %d %d %d\n",stu[0].num,stu[0].flag,stu[0].mark,stu[0].rank);

p = 2;

for(int h = 1; h < l; h++)

{ if(stu[h-1].s == stu[h].s)

{

stu[h].flag = stu[h-1].flag;

}

else

{

stu[h].flag = p;

}

p++;

}

for(i = 1; i < l; i++)

{

printf("%s %d ",stu[i].num,stu[i].flag);

printf("%d %d\n",stu[i].mark,stu[i].rank);

}

}

return 0; }

| **测试点** | **结果** | **耗时** | **内存** |
| --- | --- | --- | --- |
| 0 | 答案正确 | 2 ms | 296KB |
| 1 | 答案正确 | 2 ms | 240KB |
| 2 | 答案正确 | 2 ms | 296KB |
| 3 | 答案正确 | 44 ms | 2040KB |