**第一类**

1-1 数字个数统计1

时间限制：1S

**题目描述：**输入自然数m、n（10<m<n<=10000），分别统计m与n之间(包含端点)的 偶数个数、奇数个数、能被3整除且不能被7整除的数字个数，并求出统计数据的最大值。

**输入说明：**输入一行，包含两个自然数m和n，范围为10<m<n <=10000

**输出说明：**输出两行，第一行包含统计数字结果，偶数个数、奇数个数、能被3整除且不能被7整除数字个数，各数字之间用一个空格分隔；第二行为统计结果的最大值。

**输入样例1：**

300 500

**输出样例1：**

101 100 58

101

**输入样例2：**

200 415

**输出样例2：**

108 108 62

108

**输入样例3：**

718 1003

**输出样例3：**

143 143 82

143

#include<stdio.h>

#include<math.h>

int main()

{

int m,n;

int odd,even,d;

odd = even = d = 0;

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

int i;

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

{

if(i%2 == 0)

even ++;

else

odd ++;

if(i%3 ==0 && i%7!=0)

d ++;

}

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

int maxv;

if(even>odd)

maxv = even;

else

maxv = odd;

if(d>maxv)

maxv = d;

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

return 0;

}

1-2 数字个数统计2

时间限制：1S

**题目描述：**一个正整数n（1<n<1000），在区间[n,n2]（含端点）内统计奇数个数、偶数个数、能被4整除且不能被3整除的数字个数，并求出统计数字的最大值。

**输入说明：**输入一行，包含一个正整数n，范围为1<<n <1000。

**输出说明：**输出两行，第一行包含统计数字结果，奇数个数、偶数个数、能被4整除且不能被3整除数字个数，各数字之间用一个空格分隔；第二行为统计结果的最大值。

**输入样例1：**

5

**输出样例1：**

11 10 3

11

**输入样例2：**

18

**输出样例2：**

153 154 51

154

**输入样例3：**

20

**输出样例3：**

190 191 64

191

#include<stdio.h>

int main()

{

int n;

int odd,even,d;

odd = even = d =0;

scanf("%d",&n);

int i;

for(i=n;i<=n\*n;i++)

{

if(i%2==0)

even ++;

else

odd ++;

if(i%4==0 && i%3!=0)

d ++;

}

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

int maxv;

if(even>odd)

maxv = even;

else

maxv = odd;

if(d>maxv)

maxv = d;

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

return 0;

}

1-3 数字个数统计3

时间限制：1S

**题目描述：**输入两个三位正整数A和B，在区间[A,B]之间，或在区间[B,A]之间，完成统计任务：3的倍数数字个数、4的倍数数字个数和5的倍数且不是2的倍数的数字个数，并求出统计数字的最小值。

**输入说明：**输入一行，包含两个三位正整数A和B。

**输出说明：**输出两行，第一行包含统计数字结果，3的倍数数字个数、4的倍数数字个数、5的倍数且不是2的倍数数字个数，各数字之间用一个空格分隔；第二行为统计结果的最小值。

**输入样例1：**

100 300

**输出样例1：**

67 51 20

20

**输入样例2：**

235 121

**输出样例2：**

38 28 12

12

**输入样例3：**

577 315

**输出样例3：**

88 66 27

27

#include<stdio.h>

int getY(int a,int b)

{

if(a<b)

return a%b;

return b%a;

}

int main()

{

int a,b;

scanf("%d %d",&a,&b);

if(a>b)

{

int temp;

temp = a;

a = b;

b = temp;

}

int i;

int d3,d4,d5;

d3 = d4 =d5 =0;

for(i=a;i<=b;i++)

{

if(i%3 == 0)

d3 ++;

if(i%4 == 0)

d4 ++;

if(i%5==0 && i%2!=0)

d5 ++;

}

int minv;

if(d3<d4)

minv = d3;

else

minv = d4;

if(d5<minv)

minv = d5;

printf("%d %d %d\n%d\n",d3,d4,d5,minv);

return 0;

}

1-4 数字个数统计4

时间限制：1S

**题目描述：**输入正整数m（0<m<=1000），求出小于m的正整数中，奇数之和、偶数之和，能被5整除不能被3整除的数字之和，并求出这些和数的最大值。

**输入说明：**输入一行，包含一个正整数m，范围为0<m<=1000。

**输出说明：**输出两行，第一行包含小于m的正整数中奇数之和、偶数之和、能被5整除不能被3整除的数字之和，各数字之间用一个空格分隔。第二行输出这些和数的最大值。

**输入样例1：**

101

**输出样例1：**

2500 2550 735

2550

**输入样例2：**

53

**输出样例2：**

676 702 185

702

**输入样例3：**

11

**输出样例3：**

25 30 15

30

#include<stdio.h>

int getdiv(int a,int b)

{

if(a>b)

return a/b;

return b/a;

}

int main()

{

int m;

scanf("%d",&m);

int odds,evens,ds;

odds = evens = ds = 0;

int i;

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

{

if(i%2==0)

evens += i;

else

odds += i;

if(i%5==0 && i%3!=0)

ds += i;

}

printf("%d %d %d\n",odds,evens,ds);

int maxv;

if(evens>odds)

maxv = evens;

else

maxv = odds;

if(ds>maxv)

maxv = ds;

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

return 0;

}

1-5 数字个数统计5

时间限制：1S

**题目描述：**输入自然数n（10<=n<=1000），分别统计n与5n之间(包含端点)奇数个数、偶数个数、能被3整除且不能被5整除的数字个数，并求出各统计结果的最小值。

**输入说明：**输入一行，包含自然数n，范围为10<=n<=1000。

**输出说明：**输出两行，第一行包含统计数字结果，奇数个数、偶数个数、能被3整除且不能被5整除数字个数，各数字之间用一个空格分隔；第二行为各统计结果的最小值。

**输入样例1：**

300

**输出样例1：**

600 601 320

320

**输入样例2：**

200

**输出样例2：**

400 401 214

214

**输入样例3：**

718

**输出样例3：**

1436 1437 765

765

#include<stdio.h>

int main()

{

int n;

scanf("%d",&n);

int i;

int odd,even,d3;

odd = even = d3 = 0;

for(i=n;i<=5\*n;i++)

{

if(i%2==0)

even ++;

else

odd ++;

if(i%3==0 && i%5!=0)

d3 ++;

}

printf("%d %d %d\n",odd,even,d3);

int minv;

if(even<odd)

minv = even;

else

minv = odd;

if(d3<minv)

minv = d3;

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

return 0;

}

1-6 数字个数统计6

时间限制：1S

**题目描述：**输入两个整数m、n（100<m<n<=10000），查找区间[m，n](包含端点)内，奇数个数、偶数个数、能被7整除且不能被3整除的数字个数，并求出各统计结果的最大值。

**输入说明：**输入一行，包含两个整数m、n，范围为100<m<n <=10000。

**输出说明：**输出两行，第一行包含统计数字结果，奇数个数、偶数个数、能被7整除且不能被3整除数字个数，各数字之间用一个空格分隔；第二行为各统计结果的最大值。

**输入样例1：**

241 502

**输出样例1：**

131 131 25

131

**输入样例2：**

106 2000

**输出样例2：**

947 948 180

948

**输入样例3：**

1300 1810

**输出样例3：**

255 256 48

256

#include<stdio.h>

int getsub(int a,int b)

{

if(a>b)

return a-b;

return b-a;

}

int main()

{

int m,n;

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

int i;

int odd,even,d7;

odd = even = d7 = 0;

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

{

if(i%2==0)

even ++;

else

odd ++;

if(i%7==0 && i%3!=0)

d7 ++;

}

printf("%d %d %d\n",odd,even,d7);

int maxv;

if(even>odd)

maxv = even;

else

maxv = odd;

if(d7>maxv)

maxv = d7;

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

return 0;

}

**第二类**

1.数组排序

**题目描述：**从键盘输入一个整数a与一个长度为10的整数数组，在数组中查找是否有这个元素，如果有，将该数从数组中删除，并将最终数组从小到大输出；如果没有，则直接输出从小到大排序后的数组。

**输入格式**

第一行输入1个整数，即需要查找的数。

第二行输入10个整数（数值范围0-1000），以空格间隔。

**输出格式**

输出一行排序后的整数值，以空格间隔。

**输入样例1**

5

10 9 8 7 6 5 4 3 2 1

**输出样例1**

1 2 3 4 6 7 8 9 10

**输入样例2**

10

9 8 7 6 5 4 3 2 1 0

**输出样例2**

0 1 2 3 4 5 6 7 8 9

#include<stdio.h>

void sort(int x[],int n)

{

int i,j;

for(i=0;i<n-1;i++)

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

if(x[j]<x[i])

{

int temp;

temp = x[i];

x[i] = x[j];

x[j] = temp;

}

}

int main()

{

int a,b[10],b1[10];

scanf("%d",&a);

int i,j=0;

for(i=0;i<10;i++)

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

sort(b,10);

for(i=0;i<10;i++)

{

if(a != b[i])

{

b1[j] = b[i];

j ++;

}

}

if(j < 10)

{

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

printf("%d ",b1[i]);

printf("\n");

}

else

{

for(i=0;i<10;i++)

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

printf("\n");

}

return 0;

}

5.字符串处理1

**题目描述：**输入由数字和字符构成的字符串（不包含空格），将字符串中符合十六进制数据格式的数字和字符按照其对应的十进制数值进行累加，并输出累加结果，如果字符串中不含有任何满足十六进制格式的字符，则输出结果NO。

**输入格式**

输入一行字符串，字符串c的长度不超过50。

**输出说明**

输出一个整数，表示十进制的累加值。

**输入样例1**

3A5cH5o

**输出样例1**

35

**输入样例2**

GjoniuJk

**输出样例2**

NO

#include<stdio.h>

#include<string.h>

char ox[22]={'0','1','2','3','4','5','6','7','8','9','A','a','B','b','C','c','D','d','E','e','F','f'};

int dx[22]={0,1,2,3,4,5,6,7,8,9,10,10,11,11,12,12,13,13,14,14,15,15};

int find(char ch,char s[],int n)

{

int i;

for(i=0;i<22;i++)

{

if(ch == s[i])

return i;

}

return -1;

}

int main()

{

char str[51];

scanf("%s",str);

int sum = 0;

int len = strlen(str);

int i,pos,flag=0;

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

{

pos = find(str[i],ox,22);

if(pos>=0)

{

flag = 1;

sum += dx[pos];

}

}

if(flag)

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

else

printf("NO\n");

return 0;

}

3. 数组替换

**题目描述：**随机输入长度为n的一维数组(每个数字不相同)，找出其中的最大值和最小值，并求出最大值和最小值的最小公倍数和最大公约数，用最小公倍数替换原数组中的最大值，最大公约数替换原数组中的最小值，然后输出新的数组。

**输入格式**

第一行输入1个整数n(1<n≤20)，表示随机数组的长度。

第二行输入n个整数（数值范围1~1000），以空格间隔。

**输出格式**

在一行上输出处理后的数组，数组元素之间以空格间隔。

**输入样例**

10

6 8 2 3 7 9 10 15 25 4

**输出样例**

6 8 1 3 7 9 10 15 50 4

#include<stdio.h>

int gcd(int a,int b)

{

int i;

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

{

if(a%i==0 && b%i==0)

return i;

}

}

int times(int a,int b)

{

int i;

for(i=b;i<=a\*b;i++)

{

if(i%a==0 && i%b==0)

return i;

}

}

int main()

{

int n,a[20];

int maxv,maxi,minv,mini;

scanf("%d",&n);

int i;

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

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

maxv = minv = a[0];

maxi = mini = 0;

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

{

if(a[i]>maxv)

{

maxv = a[i];

maxi = i;

}

if(a[i]<minv)

{

minv = a[i];

mini = i;

}

}

a[mini] = gcd(minv,maxv);

a[maxi] = times(minv,maxv);

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

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

return 0;

}

6.字符串处理2

**题目描述：**任意输入一个字母x,并输入任意字符串（不包含空格），如果该字符串中包含字符x，则将字符串中满足开始与结束条件之间的字符串输出，如果该字符串中不含有字符x，则输出NO。

开始条件为：第一次出现字母x（不区分大小写）；

结束条件为：第二次出现字母x（不区分大小写）或者出现字符串结尾符

**输入格式**

输入的第一行包括一个字符x。

输入的第二行为任意字符串，长度不超过100。

**输出格式**

输出满足条件的字符串，或者输出NO。

**输入样例1**

a

12asd16w1A32d1

**输出样例1**

asd16w1A

**输入样例2**

C

vaic1351a

**输出样例2**

c1351a

**输入样例3**

c

vai1hhiubk567

**输出样例3**

NO

#include<stdio.h>

#include<string.h>

int main()

{

char x,X,s[101],s1[101]={0};

scanf("%c",&x);

scanf("%s",s);

int i,j;

if(x>='a' && x<='z')

X = 'A' + x - 'a';

else

X = 'a' + x - 'A';

int len = strlen(s);

int count=0;

int index[2];

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

{

if(s[i]==x || s[i]==X)

{

index[count] = i;

count ++;

if(count==2)

break;

}

}

if(count==2)

{

j = 0;

for(i=index[0];i<=index[1];i++)

{

s1[j] = s[i];

j ++;

}

puts(s1);

}

else if(count==0)

printf("NO\n");

else

{

j = 0;

for(i=index[0];s[i];i++)

{

s1[j] = s[i];

j ++;

}

puts(s1);

}

return 0;

}

2.数组重塑

**题目描述：**从键盘任意输入一个数字n(0<n<10)，再任意输入长度为n\*n的一串数字,将其重塑为一个n阶方阵（即行数和列数都为n的二维数组），重塑后的方阵按照先列后行从小到大的顺序进行输出（见样例）

**输入格式**

第一行输入一个整数n(0<n<10)。

第二行输入n\*n个整数，数字间用空格间隔开。

**输出格式**

输出n行，每行n个数据，数据以空格间隔。

**输入样例1**

3

6 3 9 7 5 1 15 8 10

**输出样例1**

1 6 9

3 7 10

5 8 15

**输入样例2**

3

12 121 359 63 127 15 215 10 111

**输出样例2**

10 63 127

12 111 215

15 121 359

#include<stdio.h>

int main()

{

int n,a[100];

scanf("%d",&n);

int i,N;

N = n\*n;

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

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

int j;

for(i=0;i<N-1;i++)

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

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

{

int temp;

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

int b[10][10];

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

{

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

{

b[i][j] = a[i+j\*n];

printf("%d ",b[i][j]);

}

printf("\n");

}

return 0;

}

4.数组折叠

**题目描述：**从键盘任意输入两个数字n(0<n≤6)和m(0<m≤n)，n为输入数据长度参数，m为折叠次数。然后任意输入长度为2^n(2的n次幂) 的一串数字a0，a1 ,…, a2^n-1并对该串数字进行m次折叠，最终将折叠后的数据进行输出。

数据串一次折叠将数据减少一半，规则如下：

a0+a2^n -1 , a1+a2^n -2, a2+a2^n -3 … a2^(n-1)-1+a2^(n-1)

后续每次折叠都在前一次折叠后的数组基础上使用同样的规则进行折叠。

**输入格式**

第一行输入两个整数n和m，两数用空格间隔开。

第二行输入2^n个整数，数字间用空格间隔开。

**输出格式**

输出处理后的结果，数字间用空格间隔开。

**输入样例1**

3 1

5 9 3 7 6 12 8 2

**输出样例1**

7 17 15 13

**输入样例2**

3 2

5 9 3 7 6 12 8 2

**输出样例2**

20 32

#include<stdio.h>

int main()

{

int n,m;

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

int len=1,i;

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

len \*= 2;

int a[64];

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

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

int len1,A[32];

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

{

len1 = len/2;

for(i=0;i<len1;i++)

A[i] = a[i] + a[len-1-i];

for(i=0;i<len1;i++)

a[i] = A[i];

len = len1;

}

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

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

return 0;

}

**第三类**

**题目描述：**键盘输入m×n维矩阵A，令其转置矩阵为B，求乘积矩阵C=A×B，并将C矩阵输出。

说明：把矩阵的行换成列得到的新矩阵即为转置矩阵。若A=(aij)是m×s矩阵，B=(bij)是s×n矩阵，则它们的乘积矩阵C=(cij)是一个m×n矩阵，且计算规则为：



**输入说明：**输入为m+1行，第一行是两个整数m和n（1≤m, n≤10），表示矩阵A的行数和列数，接下来m行，每行n个整数，表示构成矩阵A的m\*n个元素，各个整数之间用空格分隔。

**输出说明：**输出C矩阵。

**输入样例：**

2 3

1 2 3

4 5 6

**输出样例：**

14 32

32 77

#include<stdio.h>

int main()

{

int m,n;

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

int a[m][n],b[n][m];

int c[10][10]={0};

int i,j;

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

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

{

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

b[j][i] = a[i][j];

}

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

for(j=0;j<m;j++)

{

int k;

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

c[i][j] += a[i][k]\*b[k][j];

}

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

{

for(j=0;j<m;j++)

printf("%d ",c[i][j]);

printf("\n");

}

return 0;

}

**题目描述：**键盘输入m×n维矩阵，对其各行元素从大到小排序，输出各行排序后的矩阵。

**输入说明：**输入第一行为两个整数m和n（1≤m, n≤10），表示矩阵A的行数和列数，其后输入m行，每行n个数据，数据间以空格分隔。

**输出说明：**

输出经过各行排序后的矩阵。

共m行，每行n个整数，整数之间用空格分隔。

**输入样例：**

2 3

1 2 3

4 5 6

**输出样例：**

3 2 1

6 5 4

#include<stdio.h>

int main()

{

int m,n,a[10][10];

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

int i,j;

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

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

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

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

{

int k,l;

for(k=0;k<n-1;k++)

for(l=k+1;l<n;l++)

{

if(a[i][l]>a[i][k])

{

int temp;

temp = a[i][k];

a[i][k] = a[i][l];

a[i][l] = temp;

}

}

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

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

printf("\n");

}

return 0;

}

**题目描述：**给定二维整型数组，搜索特定数据在每行出现的次数和位置，以及哪一行搜索到的次数最多。

**输入说明：**输入第一行是两个整数m，n（1≤m,n≤200），分别表示矩阵的行数和列数；其后m行，每行n个整数，以空格间隔，表示m×n的二维整型数组；最后一行是待搜索的整数key（≤100000）。

**输出说明：**按矩阵行的顺序输出m行，给出每行的搜索情况。首先输出本行中key值出现的次数，然后按照先后顺序依次输出key出现的位置（位置从0开始计数），若次数为0，则输出0即可。最后一行输出key值出现次数最多的行标（行标从0开始计数，若有多行出现key值次数相等，输出序号小的行标），若每行均未搜索到，输出no。

**输入样例1：**

2 10

0 1 2 3 4 4 6 7 8 9

1 2 3 4 5 6 5 4 3 2

5

**输出样例1：**

0

2 4 6

1

**输入样例2：**

2 10

1 2 3 4 5 6 5 4 3 2

0 1 2 3 4 4 6 7 8 9

10

**输出样例2：**

0

0

no

#include<stdio.h>

int main()

{

int m,n,key;

int a[200][200];

int i,j;

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

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

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

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

scanf("%d",&key);

int index[200];

int maxC=0,maxi=0;

int flag = 0;

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

{

int count=0;

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

{

if(key == a[i][j])

{

index[count] = j;

count ++;

}

}

if(count>maxC)

{

maxC = count;

maxi = i;

}

if(count == 0)

printf("0\n");

else

{

flag = 1;

printf("%d ",count);

for(int k=0;k<count;k++)

printf("%d ",index[k]);

printf("\n");

}

}

if(flag)

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

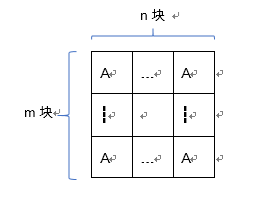
else

printf("no\n");

return 0;

}

**题目描述：**输入矩阵A，并输入两个整数m和n，对矩阵A进行m×n块复制，生成新的矩阵并输出，该过程示意图如下：



**输入说明：**输入分三部分，第一行输入两个整数a和b（1≤a, b≤10），表示矩阵A的行数和列数，第二部分是矩阵A的a×b个元素，第三部分是两个整数m和n（1≤m, n≤5），表示矩阵A分别沿着行方向和列方向被复制的次数。

**输出说明：**输出为经过复制后的矩阵。

**输入样例：**

2 3

1 2 3

4 5 6

3 2

**输出样例：**

1 2 3 1 2 3

4 5 6 4 5 6

1 2 3 1 2 3

4 5 6 4 5 6

1 2 3 1 2 3

4 5 6 4 5 6

#include<stdio.h>

int main()

{

int a,b,m,n;

int A[10][10],AA[50][50];

scanf("%d %d",&a,&b);

int i,j;

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

for(j=0;j<b;j++)

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

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

int k,l;

for(k=0;k<m;k++)

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

{

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

for(j=0;j<b;j++)

AA[i+a\*k][j+b\*l] = A[i][j];

}

for(i=0;i<m\*a;i++)

{

for(j=0;j<n\*b;j++)

printf("%d ",AA[i][j]);

printf("\n");

}

return 0;

}

**题目描述：**输入维度为m（m为正偶数）的方阵，将其沿中心点所在的水平线和垂直线平均分成4块，每块都是(m/2)\*(m/2)的方阵，然后将其沿中心点进行分块顺时针90度旋转后输出。

**输入说明：**输入为m+1行，第一行为正偶数m（≤10），表示方阵的维度，接下来m行，每行m个整数，表示m\*m个方阵元素，整数之间用空格分隔。

**输出说明：**

输出顺时针90度分块旋转后的方阵。一共输出m行，每行m个整数，整数之间用空格分隔。

**输入样例：**

4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

**输出样例：**

9 10 1 2

13 14 5 6

11 12 3 4

15 16 7 8

#include<stdio.h>

int main()

{

int m,a[10][10];

scanf("%d",&m);

int i,j;

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

for(j=0;j<m;j++)

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

int n = m/2;

int b[n][n];

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

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

b[i][j] = a[i][j];

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

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

a[i][j] = a[i+n][j];

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

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

a[i][j] = a[i][j+n];

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

for(j=n;j<m;j++)

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

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

for(j=n;j<m;j++)

a[i][j] = b[i][j-n];

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

{

for(j=0;j<m;j++)

{

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

}

printf("\n");

}

return 0;

}

**题目描述：**输入m行3列的整数矩阵，若将每行的3个数据作为三角形的三条边长，判断其是否能构成三角形，如果可以，计算该三角形面积的平方并取整(向上取整)，并按照从大到小的顺序输出此取整后的面积平方及该三角形所在的行号（行号从0开始计数），如面积取整后相等，则行数小的优先输出。如无满足条件的行，输出no。

（注：三边长为a,b,c的三角形的面积平方等于p\*(p-a)\*(p-b)\*(p-c)，其中p为周长的一半）

**输入说明：**输入分两部分，第一部分是一个整数m（1≤m≤30），表示矩阵的行数，第二部分是构成矩阵的m×3个整数；

**输出说明：**输出若干行，每一行两个整数，第一个整数表示三角形面积的平方取整，第二个数字表示该三角形所在的行号；

**输入样例1：**

3

2 3 4

1 2 3

5 12 13

**输出样例1：**

900 2

8 0

**输入样例2：**

3

1 2 3

4 5 9

11 2 9

**输出样例2：**

no

#include<stdio.h>

int main()

{

int m;

scanf("%d",&m);

int i,j;

int area[30]={0},index[30];

int a,b,c,count=0;

double p;

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

{

scanf("%d %d %d",&a,&b,&c);

if(a+b>c && a+c>b && b+c>a)

{

p = (a+b+c)/2.0;

area[count] = (int)(p\*(p-a)\*(p-b)\*(p-c));

index[count] = i;

count ++;

}

}

for(i=0;i<count-1;i++)

for(j=i+1;j<count;j++)

{

if(area[j]>area[i])

{

int temp;

temp = area[i];

area[i] = area[j];

area[j] = temp;

temp = index[i];

index[i] = index[j];

index[j] = temp;

}

}

if(count>0)

{

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

printf("%d %d\n",area[i],index[i]);

}

else

printf("no\n");

return 0;

}

**第四类**

**题目描述**

一个合法的身份证号码由地区、日期编号和顺序编号共17位再加1位校验码M组成。校验码的计算规则如下：

先对前17位数字加权求和，权值分配为：{7, 9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2}，如记前17位数字为，权值记为，则计算加权求和sum的公式为：



然后将计算的和对11取模得到值Z；最后按照以下关系对应Z值与校验码M的值：

Z：0 1 2 3 4 5 6 7 8 9 10

M：1 0 X 9 8 7 6 5 4 3 2

现在给定一些身份证号码，请验证校验码的有效性，并输出有问题的号码。

**输入格式**

第一行给出正整数N（≤100）是输入的身份证号码的个数。随后N行，每行给出一个18位的身份证号码。

**输出格式**

按照输入的顺序每行输出一个有问题的身份证号码。这里并不校验前17位是否合法，只检查前17位是否全为数字且最后一位校验码是否准确。如果所有号码都正常，则输出“All passed”。

**输入样例1**

3

320124198808240056

110108196711301866

12010X198901011234

**输出样例1**

110108196711301866

12010X198901011234

**输入样例2**

2

320124198808240056

132440197706181612

**输出样例2**

All passed

#include<stdio.h>

int main()

{

int n;

char id[20];

int w[17]={7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};

char m[15]={'1','0','X','9','8','7','6','5','4','3','2'};

scanf("%d",&n);

int i,j,flag=1;

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

{

scanf("%s",id);

int z,sum = 0;

for(j=0;j<17;j++)

{

if(id[j]<'0' || id[j]>'9')

break;

sum += (id[j]-'0')\*w[j];

}

if(j<17)

{

puts(id);

flag = 0;

}

else

{

z = sum%11;

if(m[z] != id[17])

{

puts(id);

flag = 0;

}

}

}

if(flag)

printf("All passed\n");

return 0;

}

**题目描述**

软件工程重点实验室有研究生45人，每天第一个到实验室的人要把门打开，最后一个离开的人要把门关好。现有一堆杂乱的实验室签到、签离记录，请根据记录找出当天开门和关门的人。（没有同时签到或同时签离）

**输入格式**

第一行给出正整数N（≤100）是当天的记录条数。随后N行，每行为一个学生的签到记录，包含学生学号、签到时间、签离时间，数据之间以空格分隔。其中学号为10位的数字，时间格式为：HH:MM:SS，时间数据保证合法。

**输出格式**

在一行输出开门和关门的学生学号，以空格间隔。

**输入样例**

3

2019031001 15:30:28 17:00:10

2019031005 08:00:00 11:25:25

2019031059 21:45:00 21:58:40

**输出样例**

2019031005 2019031059

#include<stdio.h>

#include<string.h>

typedef struct

{

int h,m,s;

}Time;

Time GetTime(char str[])

{

Time t;

t.h = (str[0]-'0')\*10+(str[1]-'0');

t.m = (str[3]-'0')\*10+(str[4]-'0');

t.s = (str[6]-'0')\*10+(str[7]-'0');

return t;

}

int isGreat(Time t1,Time t2)

{

if(t1.h > t2.h)

return 1;

if(t1.h == t2.h)

{

if(t1.m > t2.m)

return 1;

if(t1.m == t2.m)

{

if(t1.s > t2.s)

return 1;

}

}

return 0;

}

int main()

{

int n;

int i;

char id[15];

char str1[10],str2[10];

Time maxt = {0,0,0};

Time mint = {24,60,60};

Time cur\_int,cur\_outt;

char id\_in[15],id\_out[15];

scanf("%d",&n);

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

{

scanf("%s%s%s",id,str1,str2);

cur\_int = GetTime(str1);

cur\_outt = GetTime(str2);

if(isGreat(mint,cur\_int))

{

strcpy(id\_in,id);

mint = cur\_int;

}

if(isGreat(cur\_outt,maxt))

{

strcpy(id\_out,id);

maxt = cur\_outt;

}

}

printf("%s %s\n",id\_in,id\_out);

return 0;

}

**题目描述**

某城镇进行人口普查，得到了全镇居民的生日。请编写程序，找出镇上最年长和最年轻的人。（这里假设镇上没有超过200岁的老人，假设今天是2021年1月9日，所以超过200岁的生日和未出生的生日都是不合理的，应该被过滤掉）

**输入格式**

第一行给出正整数N（≤105）表示普查人数。随后N行，每行为一个人的姓名（由不超过10个英文字母组成的字符串）以及按“yyyy/mm/dd”（即年/月/日，日期数据均合法）格式给出的生日。题目保证最年长和最年轻的人没有并列。

**输出格式**

输出分三行，分别表示有效生日的个数、最年长人和最年轻人的姓名。如果没有有效生日，则输出0。

**输入样例1**

3

John 2001/05/12

Tom 1910/09/06

Ann 2121/01/30

**输出样例1**

2

Tom

John

**输入样例2**

3

John 2022/05/12

Tom 1810/09/06

Ann 2121/02/10

**输出样例2**

0

#include<stdio.h>

typedef struct

{

int y,m,d;

}Date;

Date left = {1821,1,9},right = {2021,1,9};

typedef struct

{

char name[15];

Date birthday;

}person;

int isGreat(Date d1,Date d2)

{

if(d1.y<d2.y)

return 1;

else if(d1.y == d2.y)

{

if(d1.m<d2.m)

return 1;

else if(d1.m == d2.m)

{

if(d1.d<d2.d)

return 1;

}

}

return 0;

}

int isRight(person p)

{

Date temp;

temp = p.birthday;

if(isGreat(temp,left))

return 0;

else if(isGreat(right,temp))

return 0;

return 1;

}

int main()

{

int n;

scanf("%d",&n);

person temp,old,young;

old.birthday = right;

young.birthday = left;

int i,count=0;

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

{

scanf("%s %d/%d/%d",temp.name,&temp.birthday.y,&temp.birthday.m,&temp.birthday.d);

if(isRight(temp))

{

count ++;

if(isGreat(temp.birthday,old.birthday))

old = temp;

if(isGreat(young.birthday,temp.birthday))

young = temp;

}

}

if(count == 0)

printf("0\n");

else

printf("%d\n%s\n%s\n",count,old.name,young.name);

return 0;

}

**题目描述**

有N个结点围成一个圈，每个结点每次只能移动到与它左右相邻的结点，给出每个结点的编号和坐标，然后给出若干询问，每个询问给出两个数字A和B（1≤A, B≤N，均表示结点的编号），求从A号结点到B号结点的最短距离。（具体见样例1示意图解释）

说明：若p1和p2两点的坐标分别为(x1,y1)和(x2,y2)，则这两点间的距离



**输入格式**

第一行给出正整数N（3≤N≤105）表示结点个数。随后N行表示按编号顺序排列的每个结点的信息，结点信息包括结点编号num（1≤num≤N）和坐标（坐标值为两个整数，分别表示结点的x和y坐标）。接下来给出若干行询问，每行询问由两个整数A和B组成，以空格间隔，当输入两个数字0时结束循环。

**输出格式**

输出若干行，针对每行询问，输出两个节点间的最短距离，数值保留小数点后两位。

**输入样例1**

4



1 5 5

2 -5 5

3 -5 -5

4 5 -5

1 3

1 4

0 0

**输出样例1**

20.00

10.00

图1 样例1示意图

**样例1示意图解释：**

如图1所示，样例1共有4个结点，编号分别为1、2、3、4，后面括号里给出了结点坐标。下面总共有2个询问：

1 3：从1号结点到3号结点，路径可以选择1🡪2🡪3或者1🡪4🡪3，距离均为20；

1 4：从1号结点到4号结点，最短路径应为1🡪4，最短距离为10。

**输入样例2**

5

1 0 12

2 7 7

3 6 2

4 -5 2

5 -6 7

1 3

2 5

4 1

0 0

**输出样例2**

13.70

16.41

12.91

#include<stdio.h>

#include<math.h>

typedef struct

{

int num;

int x;

int y;

}Node;

double GetDistance(Node n1,Node n2)

{

double d;

int dx,dy;

dx = n1.x-n2.x;

dy = n1.y-n2.y;

d = sqrt(dx\*dx + dy\*dy);

return d;

}

int main()

{

int n,a,b;

scanf("%d",&n);

int i;

Node nd[n+1];

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

{

scanf("%d %d %d",&nd[i].num,&nd[i].x,&nd[i].y);

}

double sum = 0;

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

{

sum += GetDistance(nd[i],nd[i+1]);

}

sum += GetDistance(nd[n],nd[1]);

while(1)

{

scanf("%d %d",&a,&b);

if(a==0 && b==0)

break;

if(a>b)

{

int temp;

temp = a;

a = b;

b = temp;

}

int i;

double dis=0,dis1;

for(i=a;i<b;i++)

{

dis += GetDistance(nd[i],nd[i+1]);

}

dis1 = sum-dis;

if(dis<dis1)

printf("%.2f\n",dis);

else

printf("%.2f\n",dis1);

}

return 0;

}

**题目描述**

有N位同学非常喜欢一起玩游戏，他们决定选择其中一个人所在的地点玩游戏，但大家都很懒，不想走很远，他们想知道选哪个人的坐标作为集合点可以让大家走的总路程最短。于是请你来替他们解决这个问题，作为奖励，他们0会告诉你期末考试的题目，没错，就是这道题。

说明：若p1和p2两点的坐标分别为(x1,y1)和(x2,y2)，则这两点间的距离



**输入格式**

第一行给出正整数N（3≤N≤105）表示人数。随后N行表示按编号顺序排列的每位同学的信息，信息包括学生编号num（1≤num≤N）和坐标（坐标值为两个整数，分别表示该学生所在地点的x和y坐标）。

**输出格式**

输出一行，包括集合点所属同学的编号，以及到该集合点大家需要走的总路程（保留小数点后2位），数据之间以空格分隔。如果满足条件的集合点不止一个，输出编号最大的那个集合点。

**输入样例1**

5

1 0 12

2 7 7

3 6 2

4 -5 2

5 -6 7

**输出样例1**

5 38.91

**输入样例2**

4

1 5 5

2 -5 5

3 -5 -5

4 5 -5

**输出样例2**

4 34.14

#include<stdio.h>

#include<math.h>

typedef struct

{

int num;

int x;

int y;

}Node;

double GetDistance(Node n1,Node n2)

{

double d;

int dx,dy;

dx = n1.x-n2.x;

dy = n1.y-n2.y;

d = sqrt(dx\*dx + dy\*dy);

return d;

}

double GetSum(Node nd[],int n,int i)

{

double sum = 0;

int j;

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

sum += GetDistance(nd[j],nd[i]);

return sum;

}

int main()

{

int n,a,b;

scanf("%d",&n);

int i;

Node nd[n+1];

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

{

scanf("%d %d %d",&nd[i].num,&nd[i].x,&nd[i].y);

}

double curD,minD=0;

int minIndex = 1;

minD = GetSum(nd,n+1,1);

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

{

curD = GetSum(nd,n+1,i);

if(curD <= minD)

{

minD = curD;

minIndex = i;

}

}

printf("%d %.2f\n",minIndex,minD);

return 0;

}

**题目描述**

有54张牌，牌面分别为2、3、4、5、6、7、8、9、10、J、Q、K、A、W。其中W有2张，包含两种花色，分别为red和black，其余牌面每种均有4张，包含四种花色，分别为heart、spade、club、diamond。接下来执行洗牌操作，将牌的位置改变为指定位置。例如有5张牌，排列顺序为：heart-3、spade-K、club-10、diamond-A、red-W，给定洗牌操作序列{4,2,5,3,1}，即把heart-3放到4号位、spade-K放到2号位、club-10放到5号位、diamond-A放到3号位、red-W放到1号位，于是排序就变成了red-W、spade-K、diamond-A、heart-3、club-10。如果要将上面的洗牌操作执行第2次，则最后的洗牌结果为club-10、spade-K、heart-3、red-W、diamond-A。

**输入格式**

第一行给出正整数N（1≤N≤54）表示纸牌的数量。随后N行表示N张牌的初始排列信息，信息包括牌面和花色，以空格间隔。

接下来一行给出一个正整数K（≤100），表示洗牌操作次数；最后一行给出洗牌操作序列，包含N个正整数，正整数取值范围在1到54之间，以空格间隔。

**输出格式**

输出N行，给出N张牌的最后排列结果，每行给出一张牌的牌面和花色，以空格间隔。

**输入样例**

5

3 heart

K spade

10 club

A diamond

W red

2

4 2 5 3 1

**输出样例**

10 club

K spade

3 heart

W red

A diamond

#include<stdio.h>

typedef struct

{

char face[3];

char suit[8];

}Card;

int main()

{

int N;

Card cards\_start[55],cards\_end[55];

int i;

scanf("%d",&N);

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

{

scanf("%s %s",&cards\_start[i].face,&cards\_start[i].suit);

}

int K,pos[55];

scanf("%d",&K);

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

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

for(int step=0;step<K;step++)

{

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

cards\_end[pos[i]] = cards\_start[i];

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

cards\_start[i] = cards\_end[i];

}

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

printf("%s %s\n",cards\_start[i].face,cards\_start[i].suit);

return 0;

}