**A Problem Description**

jhljx最近学了二维数组，决定用二维数组来打印一些图形。他找到了一块正方形的小木板，木板上有一些小方格。他决定把一些数字填入到这些小方格里。但是他喜欢转圈圈，他喜欢让这些数字螺旋的排列在正方形的小方格里。听起来很有趣，希望你们来帮助他吧。

**Input**

输入多组测试数据。  
每组测试数据为一个数字n。(1<=n<=1000)

**Output**

输出这个正方形的图案，并且计算出主对角线(左上角到右下角)上元素的值。(输出结果详见样例)

**Sample Input**

3

5

**Sample Output**

1 2 3

8 9 4

7 6 5

15

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

73

这道题可以定义一个变量使其连续++，控制这个变量按照螺旋形旋转。

显然要分成向右，向下，向左，向上四步。

可以写这样的四个循环（难点是循环的起点和次数，要在草纸上多做实验），在外面套个大循环(大循环是一直循环到中心停止)。

参考代码：

#include<iostream>

using namespace std;

int a[1001][1001]= {0};

int main()

{

int n;

while(cin>>n)

{

int m,i,j,k=0;//m为大循环的次数（要分n的奇偶）。

if(n%2==0)//计算m的数值。

m=n/2;

else

m=n/2+1;

for(i=0; i<=m-1; i++)大循环。

{

for(j=i; j<=n-i-1; j++)向右的小循环。

{

k++;

a[i][j]=k;

}

for(j=i+1; j<=n-i-1; j++)向下的小循环。

{

k++;

a[j][n-i-1]=k;

}

for(j=n-i-2; j>=i; j--)向左的小循环。

{

k++;

a[n-i-1][j]=k;

}

for(j=n-i-2; j>=i+1; j--)向上的小循环。

{

k++;

a[j][i]=k;

}

}

for(int ii=0; ii<n; ii++)打印出来。

{

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

{

cout<<a[ii][jj]<<" ";

}

cout<<endl;

}

long long sum=0;

for(int ii=0; ii<n; ii++)//计算对角线的和。

{

sum+=a[ii][ii];

}

cout<<sum<<endl;//输出sum。

}

}

**B Problem Description**

jhljx最近喜欢上下楼梯玩耍，他找到了这样一种楼梯。

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

...

这种楼梯好神奇吖。。  
他希望你们用会二维数组。jhljx温馨提醒：不要耍小聪明来水过。

**Input**

输入多组数据。  
每组数据第一行为一个整数n(1<=n<=100),表示这个楼梯有多少层。第二行为一个整数k(1<=k<=100)，表示有k次访问。  
后边k行每行为两个正整数x,y。表示第x层，第y列的位置。(保证x,y在int范围内)

**Outut**

每组数据输出一行，如果a[x][y]存在，输出这个位置的数，反之输出-1。

**Sample Input**

5

5

1 1

3 2

4 3

5 3

5 7

**Sample Output**

1

2

3

6

-1

**Hint**

**请用long long类型的二维数组实现**

此题与这道题<http://acm.buaa.edu.cn:8081/contest/115/problem/A/> 很类似。

题目提示要用long long 类型的二维数组来做。

观察得知每个数等于它正上面的数和左斜上方的数的和（不存在的数定义为0）。

发现这个规律用二维数组就好做多了。

另外我发现这道题说“多组测试数据”，如果组数特别多，我们就不划算对于每个n和每个测试的k都重新计算一次数组的值（可能会超时）。我们可以在输入n之前就把最大的n=100时的数组计算好。

在下面直接使用。

参考代码：

#include<iostream>

using namespace std;

int main()

{

long long a[101][101]={};//初始化全部为0。

a[1][1]=1;//要注意手动把第一个数赋值为1.

for(int i=1;i<=100;i++)//计算n=100时的数组的值。

{

for(int j=1;j<=100;j++)

{

if(i==1&&j==1){}

else

a[i][j]=a[i-1][j-1]+a[i-1][j];

}

}

int n;

while(cin>>n)

{

int x,y,k;

cin>>k;

while(k--)

{

cin>>x>>y;

if(x>n||y>x)

cout<<"-1"<<endl;

else

cout<<a[x][y]<<endl;

}

}

}

# C Problem Description

jhljx学习了数组的冒泡排序,给你n个数，请你按照从大到小的顺序排列。

# Input

输入多组数据。  
每组数据一行为一个正整数n(1<=n<=1000)。  
接着输入n个正整数。(保证数字在int范围内)

# Output

每组数据输出两行。保证每组的n个数从大到小排列，并输出冒泡排序中相邻两个数交换的次数。

# Sample Input

5

6 -1 3 7 10

# Sample Output

10 7 6 3 -1

8

# Hint

**请用冒泡排序实现。禁用STL。**

此题比较简单。至于STL百度一下吧。

参考代码：

#include<iostream>

using namespace std;

int main()

{

int n;

while(cin>>n)

{

int a[1001]= {0},cn=0;//cn为交换次数。

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

cin>>a[i];

for(int i=1; i<=n-1; i++)//冒泡排序，个人喜欢从1开始。

{

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

{

if(a[j]<a[j+1])//如果把“<”变为“>”就变成了从小到大排序。

{

int t=a[j];

a[j]=a[j+1];

a[j+1]=t;

cn++;//次数++。

}

}

}

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

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

cout<<endl<<cn<<endl;;

}

}

# D Problem Description

KamuiKirito遇到一只萌萌哒怪物  
  
这只萌萌哒怪物喜欢吃糖果。  
现在有n块糖果。每块糖果有质量和含糖量。  
现在这只怪物只能吃质量和为k的糖果。  
而这只怪物每次会选择单位质量含糖量最高的糖果吃掉。  
而如果这颗糖果她已经吃不下了，她就会停止并且放弃这颗糖果。  
那么她会获得多少糖分呢。

# Input

输入多组数据。  
每组数据第一行为两个整数n(1<=n<=1000),k（0<=k<=100000)表示有n块糖果,怪物最多可以吃质量和为k的糖果。  
第二行为n个正整数ai(a1,a2,……an)，代表第i块糖果的质量。  
第三行为n个正整数bi(b1,b2,……bn)，代表第i块糖果的含糖量。

# Outut

每组数据输出一行，为怪物最多能吃到的糖分。

# Sample Input

3 4  
1 2 3  
3 2 1

# Sample Output

5

此题思路比较清晰。先计算好每块糖单位质量含糖量，再降序排一下。

注意如果最后的那块糖吃一半时就饱了，那么这块糖是不能吃的。

还要注意如果都吃完了但是还没饱的情况。

参考代码：

#include<iostream>

using namespace std;

int main()

{

int n,k;

while(cin>>n>>k)

{

double a[1001]={},b[1001]={},c[1001]={};/\*注意此处都要声明为double，不要把a,b声明为整形！如果为整形，在计算过程会有偏差（光用double来接受是不行的）。\*/

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

cin>>a[i];

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

cin>>b[i];

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

c[i]=b[i]/a[i];

for(int i=1;i<=n-1;i++)//排序。

{

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

{

if(c[j]<c[j+1])

{

double t1=a[j],t2=b[j],t3=c[j];//此处要同时转换三对值。

a[j]=a[j+1];

b[j]=b[j+1];

c[j]=c[j+1];

a[j+1]=t1;

b[j+1]=t2;

c[j+1]=t3;

}

}

}

long long sum=0;//质量和。

int sum1=0;//糖分和。

bool flag=0;

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

{

sum+=a[i];

if(sum>k)//判断一下是否饱了。

{

flag=1;

for(int j=1;j<=i-1;j++)//注意此处为i-1非i！

sum1+=b[j];

break;

}

}

if(!flag)

{

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

sum1+=b[j];

}

cout<<sum1<<endl;

}

}

# E Problem Description

全能者能创造出一块他搬不动的石头吗？  
-------------------------------------------------------  
现在有若干块石头，每块石头拥有质量和价值。请输出  
单位质量价值第二大的石头的序号。

# Input

输入多组数据。  
每组数据第一行为一个整数n(2<=n<=1000),表示有n块石头。  
第二行为n个整数ai(a1,a2,……an)，代表第i个石头的价值。  
第三行为n个正整数bi(b1,b2,……bn)，代表第i个石头的质量。

# Outut

每组数据输出一行，为单位质量价值第二大的石头的序号。

# Sample Input

3  
1 2 3  
3 2 1

# Sample Output

2

# Hint

保证第一大和第二大不存在并列现象。

这道题与上一道有共同点。

重点是把最大值赋值为-1，再找出最大值。

参考代码：

#include<iostream>

using namespace std;

double a[1001]= {},b[1001]= {};//由于每次都输入，所以不必memset。

double c[1001]= {};

int main()

{

int n;

while(cin>>n)

{

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

cin>>a[i];

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

cin>>b[i];

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

c[i]=a[i]/b[i];

double max1=c[1];

int maxnum=1;

for(int i=2; i<=n; i++)//找出最大值。

{

if(c[i]>max1)

{

maxnum=i;

max1=c[i];

}

}

c[maxnum]=-1;//变为-1.

max1=c[1];

maxnum=1;

for(int i=2; i<=n; i++)//再比较一次。

{

if(c[i]>max1)

{

maxnum=i;

max1=c[i];

}

}

cout<<maxnum<<endl;

}

}

# F Problem Description

给你一升序数列，以及一组查询，查询某一特定元素是否存在于数列之中，如果存在，则输出该元素首次出现的位置，否则输出"error"。  
输出m行，每行输出内容见题目描述及样例。

# Input

多组测试数据。  
每组数据第一行为两个整数n,m(1<=n,m<=1000000)，表示数列中有n个元素以及m次查询。  
第二行包含n个正整数（1<=ai<=2000000)，用空格分隔，表示有序数列。  
接下来m行，每行一个整数，表示每次查询的元素。

# Output

每组数据输出一行。见样例。

# Sample Input

5 3  
1 2 3 4 5  
3  
5  
7

# Sample Output

3  
5  
error

当然此题要是用线性查找肯定超时。由于已经排好了序，所以不用再排序了。

可以另外开一个数组用来记录每个数对应的i。到时直接输出aa[a[i]]即可。

注意由于测试次数太多，用scanf才不会超时。

参考代码：

#include<iostream>

#include<cstring>

#include<cstdio>

using namespace std;

long long a[1000001]= {},aa[2000001]= {};//aa用来记录每个数对应的i。

int main()

{

int n,m;

while((scanf("%d %d",&n,&m))!=EOF)

{

memset(a,0,sizeof(a));//初始化。

memset(aa,0,sizeof(aa));

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

{

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

**aa[a[i]]=i;**

}

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

{

long long b;

scanf("%lld",&b);

if(aa[b]==0)

printf("error\n");

else

printf("%lld\n",aa[b]);

}

}

}