第七次上机解题报告

By——14211103 田旺

写在前面：

本次上机呢，难度的确不大。都是老师上课讲过的东西。但结果是，做的效果却不如想象中的好。我想，或许是因为宋友老师上午才讲，没有足够多的时间去消化理解练习的缘故吧，感觉还是略显生疏。不过，就个人来说，上课听讲真的很重要啊·····

A 题：jhljx 转圈圈

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
```

解题思路：

这道题难度较大，比较难想。自己画两遍后，可以感觉到，螺旋是有周期性的，每一个周期内，包含左，下，右，上四个部分，那么，就可以考虑定义四个方向变量，用来控制四个方向的数组“走向”。还有就是，这道题能说的就这么多，但具体的数据关系是需要细心的去找的。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<iomanip>
#include<cstring>
using namespace std;
const int maxn=10010;
int a[maxn][maxn];
int main()
{
	int N;
	while(cin>>N)
	{
	int n,i,j,m=1;
	int s=0;
	for(n=0;n<=(N+1)/2;n++)//计算周期循环次数。
	{
		for(j=n;j<N-n;j++)
			a[n][j]=m++;//左
		for(i=n+1;i<N-n;i++)
			a[i][N-n-1]=m++;//下
		for(j=N-n-2;j>=n;j--)
			a[N-n-1][j]=m++;//右
		for(i=N-n-2;i>n;i--)
			a[i][n]=m++;//上
	}//以上就是整个解题过程最重要的部分，至于具体的数据变化，需要自己用笔仔细找一找，不难，但比较繁琐。
	for(i=0;i<N;i++)
	{
		for(j=0;j<N;j++)
```

```
                cout<<a[i][j]<<" ";
                cout<<endl;
        }
        for(int x=0;x<=N-1;x++)
        {
            s+=a[x][x];
        }
        cout<<s<<endl;
        }//输出部分就很好理解啦。
}
```

B 题：jhljx 下楼梯

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

解题思路：

这道题还是比较简单的。稍微观察一下就可以发现，这其实就是杨辉三角。每行除了第一个和最后一个都是 1 外，剩余的中间的数都是其“肩上”的两个数的和，用数组表示就是 a[i][j]=a[i-1][j-1]+a[i-1][j]。在查询时，需要注意判断 X，Y 是否是合法的（即 x<=n&&y<=x 是否满足），最后输出 a[x][y]就行了。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<iomanip>
#include<cstring>
using namespace std;
const int maxn=200;
long long a[maxn][maxn];//注意，二维数组的大小是 maxn*maxn，所以此处的 maxn 不能太大，只要够用就行了。
int main()
{
    int n,s,x,y;//n 为层数，s 为查询次数。
    while(cin>>n)
    {
        cin>>s;
        for(int b=1;b<=s;b++)
        {
            cin>>x>>y;
            if(n<=2)
```

```
        a[1][1]=1;a[2][1]=1;a[2][2]=1;//当只有 2 层时，直接采取枚举的方式，将所有元素赋值就
行。
        if(n>=3)
        {
            a[1][1]=1;a[2][1]=1;a[2][2]=1;//这是后面运算的基础。
            for(int i=3;i<=n;i++)
            {
                a[i][1]=1;a[i][i]=1;//先把每行的开头和结尾的 1 赋值给相应数组元素。
                for(int j=2;j<=i-1;j++)
                {
                    a[i][j]=a[i-1][j-1]+a[i-1][j];//对于中间的数，就将其“肩上”数相加。
                }
            }
        }
            if(x<=n&&y<=x)
            cout<<a[x][y]<<endl;
            else
            cout<<-1<<endl;//判断输入查询的 X，Y 是否合法，并相应输出。
            memset(a,0,sizeof(0));//将数组清零是个好习惯。
        }
    }
}
```

C 题：jhljx 学排序

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。

解题思路:

冒泡排序，宋友老师上课讲过，不多说了。

参考代码:

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<iomanip>
using namespace std;
const int maxn=1000100;
int a[maxn];
int main()
{
    int n;
    while(cin>>n)
    {
        int s=0;
        for(int i=1;i<=n;i++)
        {cin>>a[i];}
        for(int round=0;round<=n-1;round++)
        {
            for(int b=1;b<=n-round-1;b++)
            {
              int x=a[b];
              if(a[b]<a[b+1])
              {a[b]=a[b+1];a[b+1]=x;s++;}//此处需要加入一个计数变量 s，用于记录进行冒泡的次数。
            }
        }
```

```
            for(int c=1;c<=n;c++)
            cout<<a[c]<<" ";
            cout<<endl;cout<<s<<endl;
        }
}
```

D 题：全能者的悖论

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。

参考代码：

```
#include<iostream>
```

```
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<iomanip>
#include<cstring>
const int maxn=1000100;
int a[maxn];int b[maxn];double c[maxn];//为了保障精确度，此处请将数组 c 的类型设为 double 类型。
using namespace std;

int main()
{
    int n;
    while(cin>>n)
    {
        int e=0,f=0;//e 和 f 待会儿用来记录最大的数组元素是数组中的第几个元素。
        for(int i=0;i<n;i++)
        cin>>a[i];
        for(int j=0;j<n;j++)
        cin>>b[j];
        for(int x=0;x<n;x++)
        c[x]=(double)a[x]/b[x];//计算并保存数组 c[x]，注意为 double 类型。
        double ma=-1;//先假设一个最大值为数组中不可能存在的-1。
        for(int v=0;v<n;v++)
        {
            if(c[v]>=ma)
             {
                 ma=c[v];
                 e=v;
             }
        }//第一遍，找出数组中的最大值，并记录下它是第几个元素。
        c[e]=-1;double second=-2;//将数组中最大的元素赋值为-1，相当于将其从数组中“清除”。同时，设置一个数组中第二大值的变量，将其赋值为数组中不可能存在的值-2.
        for(int d=0;d<n;d++)
        {
            if(c[d]>=second)
             {
                 second=c[d];
                 f=d;
             }
        }//再进行一次求最大值的操作，只不过这一次找到的最大值，已经是原来数组中的第二大值了。
        cout<<f+1<<endl;//输出相应的数组元素序号，不过在作为最后输出结果时，记住要+1.
        memset(a,0,sizeof(a));
        memset(b,0,sizeof(b));
```

```
        memset(c,0,sizeof(c));//将所有数组清零。
    }
}
```

E 题：糖果魔女

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

解题思路：

这道题，首先想到的就是需要进行一个排序，将单位质量的含糖量进行一个从大到小排序，并模拟这样一个吃糖的过程。但这个时候，还需要注意继续吃糖的条件，即是总的糖的质量不能超过 K，那么，这个时候就需要在将单位质量的含糖量进行排序时，同时还需要将对应得糖的质量和含糖量跟随着进行顺序的

变换。还需要注意的是，当单位质量的含糖量相同时，为了保证尽可能的获得更多的糖分，还需要进行对含糖量进行从大到小的排序。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
using namespace std;

int main()
{
    int n,k,i,j,temp2,total;
    long long a[2000],b[2000];
    long double c[2000],temp;
    while (cin>>n>>k)
    {
        for (i=1;i<=n;i++)
            cin>>a[i];
        for (i=1;i<=n;i++)
        {
            cin>>b[i];
            c[i]=(long double)b[i]/a[i];
        }//输入记录糖果质量的数组 a[x],记录含糖量的数组 b[x],以及同时计算单位质量的含糖量数组 c[x]。
        for (i=n-1;i>=1;i--)
          for (j=1;j<=i;j++)
            if (c[j]<c[j+1])
            {
                temp=c[j];
                c[j]=c[j+1];
                c[j+1]=temp;

                temp2=a[j];
                a[j]=a[j+1];
                a[j+1]=temp2;

                temp2=b[j];
                b[j]=b[j+1];
                b[j+1]=temp2;
            }//将数组 c 进行冒泡排序，同时，记住将另外两个数组 a，b 进行交换排序。
        for (i=n-1;i>=1;i--)
          for (j=1;j<=i;j++)
            if ((c[j]==c[j+1])&&(a[j]<a[j+1]))//注意，此处考虑的情况是，在两块糖单位质量的含糖量
```

相同时，为了保证她吃的糖的质量最大，需要选择质量大的糖吃掉。

```
            {
                temp=c[j];
                c[j]=c[j+1];
                c[j+1]=temp;

                temp2=a[j];
                a[j]=a[j+1];
                a[j+1]=temp2;

                temp2=b[j];
                b[j]=b[j+1];
                b[j+1]=temp2;
            }//排序时，同样需要将三个数组一起交换顺序。
        i=1;
        total=0;
        while ((i<=n)&&(k>0))
        {
            if (k>=a[i])
            {
                k=k-a[i];
                total+=b[i];
                i+=1;
            }
            else k=0;
        }//按照题目所述模拟吃糖的过程。
        cout<<total<<endl;
    }
}
```

F 题： 这货不是二分

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

解题思路：

首先，这货真的不是二分・・・其次，用线性查找会超时・・・那么，就不能用正常的普通思路做了。我们可以借鉴书上在讲到运用数组统计成绩时，将 a[相应的分数]++的方法来做这道题，具体的细节就写在代码注释里吧。

参考代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<iomanip>
#include<cstring>
using namespace std;
const int maxn=2000001;
int a[maxn];int b[maxn];//定义了 2 个数组，其中数组 a 是用来存储输入的数据的，至于数组 b，待会儿用到的时候再说。

int main()//为了避免超时，本题请全部采用 C 语言风格的输入输出。
{
    int weishu=0;int n,m;int sea;
    while(~scanf("%d%d",&n,&m))
    {
        int x;
```

```
            memset(a,0,sizeof(a));
            memset(b,0,sizeof(b));//将数组清零。
            for(int i=0;i<n;i++)
            {
                scanf("%d",&x);a[x]++;weishu++;//每输入一个数，就将对应的数组元素 a[输入的数值]++，来记录这个数出现了多少次。同时，为了记录每个数的出现的位置，我设置了一个变量 weishu。
                if(b[x]==0)
                {b[x]=weishu;}//数组 b 其实就是用来保存每次输入的数的位置的。如果这个数是第一次出现，那么 b[x]就为 0，那么就记录下这个数的位置变量 weishu；如果这个数不是第一次出现，那么 b[x]肯定已经被赋过一次值了，也就不满足 if 语句里的判断条件，那么这个时候就不会记录下第二次这个数出现的位置了。这就保证了输出的时候，输出的是每个数值第一次出现的位置。（真是巧妙呢・・・）
            }
            for(int j=0;j<m;j++)
            {
                scanf("%d",&sea);//输入查找的数值
                if(a[sea]==0)
                    printf("error\n");//如果 a[x]为 0 的话，证明在之前的数值输入时，没有输入相应的值（否则的话，a[x]的值最小为 1），那么就输出 error。
                if(a[sea]!=0)
                    printf("%d\n",b[sea]);//a[x]!=0，就表明在之前的输入时，输入了该值，那么输出该数值第一次出现的位置，而每个数值的位置信息是由数组 b 确定的，即输出 b[x]。
            }
            weishu=0;
        }
}
```