《一本通训练指导》一维数组

【题目描述】输出一个整数序列中与指定数字相同的数的个数。

【输入】输入包含三行：

第一行为n，表示整数序列的长度（n≤100）；第二行为n个整数，整数之间以一个空格分开；

第三行包含一个整数，为指定的数字m。

【输出】输出为n个数中与m相同的数的个数。

【输入样例】

3

2 3 2

2

【输出样例】

2

4.4-1

```
#include<iostream>//1. 与指定数字相同的数的个数
using namespace std;
int main()
{
	int i,a[1001],n,m,k=0;
	cin>>n;
	for(i=1;i<=n;i++)
		cin>>a[i];

	cin>>m;// 查找 m 这个数字
	for(i=1;i<=n;i++)
	{
		if(a[i]==m)
			k++;
	}
	cout<<k;
	return 0;
}
```

```
3
8 3 8
8

2
```

陶陶家的院子里有一棵苹果树，每到秋天树上就会结出 10 个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个 30 厘米高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。

现在已知 10 个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

输入描述：包括两行数据。第一行包含 10 个 100 到 200 之间（包括 100 和 200）的整数（以厘米为单位）分别表示 10 个苹果到地面的高度，两个相邻的整数之间用一个空格隔开。第二行只包括一个 100 到 120 之间（包含 100 和 120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度。

输出描述：包括一行，这一行只包含一个整数，表示陶陶能够摘到的苹果的数目。

输入样例：

100 200 150 140 129 134 167 198 200 111

110

样例输出：

5

```
4.4-2// 本题选自 NOIP2005 普及组复赛
#include<iostream>//2. 陶陶摘苹果
using namespace std;
int main()
{
	int a[100];
	int i,h,n=0;//h 苹果高度，n 苹果数量
	for(i=0;i<=9;i++) // 输入 10 个苹果的高度
		cin>>a[i];
	cin>>h;// 输入陶陶伸手的高度

	for(i=0;i<=9;i++)
	{
		if(h+30>=a[i])
			n++;// 如果身高加小板凳超过苹果高度，苹果数量加 1
	}
	cout<<n;
	return 0;
}
```

【题目描述】

下面是一个图书的单价表：计算概论 28.9 元/本，数据结构与算法 32.7 元/本，数字逻辑 45.6 元/本，C++ 程序设计教程 78 元/本，人工智能 35 元/本，计算机体系结构 86.2 元/本，编译原理 27.8 元/本，操作系统 43 元/本，计算机网络 56 元/本，JAVA 程序设计 65 元/本，给定每种图书购买的数量，编程计算应付的总费用。

【输入】

输入一行，包含 10 个整数（大于等于 0，小于等于 100），分别表示购买的《计算概论》、《数据结构与算法》、《数字逻辑》、《C++ 程序设计教程》、《人工智能》、《计算机体系结构》、《编译原理》、《操作系统》、《计算机网络》、《JAVA 程序设计》的数量（以本为单位）。每两个整数用一个空格分开。

【输出】

输出一行，包含一个浮点数 f，表示应付的总费用。精确到小数点后一位。

【输入样例】

1 5 8 10 5 1 1 2 3 4

【输出样例】

2140.2

4.4-3

```
1 5 8 10 5 1 1 2 3 4
2140.2
```

```
#include<iostream>//3. 计算书费
#include<cstdio>
using namespace std;
double a[10]={28.9,32.7,45.6,78,35,86.2,27.8,43,56,65};//10 种书的价格
int main()
{
	int x,i;
	double s=0;
	for(i=0;i<=9;i++)
	{
		cin>>x;// 输入购买的数量
		s=s+x*a[i];
	}
	printf("%.1f",s);
	return 0;
}
```

4.4-4// 本题选自《CCF 中学生计算机程序设计》

输入年、月、日，输出天数。

```
2000 3 15
75
```

```
#include<iostream>
using namespace std;
int main()
{
	int a[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
	int i,year,month,date,ans=0;//year 年，month 月，date 日期，ans 天数
	cin>>year>>month>>date;

	for(i=1;i<month;i++)
		ans+=a[i];// 按照每月多少天加起来

	ans+=date;// 加上日期

	if((year%4==0&&year%100!=0)||year%400==0)// 判断是否闰年
	{
		if(month>2)
			ans++;
	}
	cout<<ans;
	return 0;
}
```

【题目描述】某医院想统计一下某项疾病的获得与否与年龄是否有关，需要对以前的诊断记录进行整理，按照 0-18、19-35、36-60、61 以上（含 61）四个年龄段统计的患病人数占总患病人数的比例。

【输入】共 2 行，第一行为过往病人的数目 n（0<n<=100），第二行为每个病人患病时的年龄。

【输出】按照 0-18、19-35、36-60、61 以上（含 61）四个年龄段输出该段患病人数占总患病人数的比例，以百分比的形式输出，精确到小数点后两位。每个年龄段占一行，共四行。

【输入样例】 10

1 11 21 31 41 51 61 71 81 91

【输出样例】

20.00% 20.00% 20.00% 40.00%

4.4-5// 本题选自《一本通》

```
#include<iostream>//5. 统计年龄阶段发病率
#include<cstdio>
using namespace std;
int a[10000];
int main()
{
	int n,i;
	double a1=0,a2=0,a3=0,a4=0;//double 是很大数字的数据类型
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i];// 输入年龄的同时，同时统计出四个年龄段
		if(a[i]<=18)		a1++;
		if(a[i]>18&&a[i]<=35)	a2++;
		if(a[i]>35&&a[i]<=60)	a3++;
		if(a[i]>60)		a4++;
	}

	printf("%.2f%\n",a1/n*100);//\n 换行符
	printf("%.2f%\n",a2/n*100);//%，计算结果后加 %
	printf("%.2f%\n",a3/n*100);
	printf("%.2f%\n",a4/n*100);
	return 0;
}
```

```
10
1 11 21 31 41 51 61
71 81 91
20.00%
20.00%
20.00%
40.00%
```

校门外的树【1.6 编程基础之一维数组 06】Noip2005 普及组第 2 题

某校大门外长度为 L 的马路上有一排树，每两棵相邻的树之间的间隔都是 1 米。我们可以把马路看成一个数轴，马路的一端在数轴 0 的位置，另一端在 L 的位置；数轴上的每个整数点，即 0，1，2，……，L，都种有一棵树。一些区域的树要挪走，这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。这些区域中的树（包括区域端点处的两棵树）移走。计算将这些树都移走后，马路上还有多少棵树。

输入：第一行有两个整数 L（1 <= L <= 10000）和 M（1 <= M <= 100），L 代表马路的长度，M 代表区域的数目，L 和 M 之间用一个空格隔开。接下来的 M 行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标。

输出：包括一行，这一行只包含一个整数，表示马路上剩余的树的数目。

```
4.4-6// 选自《一本通》
#include<iostream>//6. 校门外的树
using namespace std;
int main()
{
	bool a[10001];
	int m,l,i,j,x,y,s=0;
	cin>>l>>m;//l 是区域总长度，m 是要挪走树的区域
	for(i=0;i<=l;i++) a[i]=1;// 初始化为树木都存在，L+1 棵树
	for(i=1;i<=m;i++) //m 个区域的树被挪走
	{
		cin>>x>>y;
		for(j=x;j<=y;j++)
			a[j]=0;
	}
	for(i=0;i<=l;i++)
	{
		if(a[i]==1)
			s++;
	}
	cout<<s;
	return 0;
}
```

样例输入：

```
500 3
150 300
100 200
470 471
```

样例输出：

```
298
```

【题目描述】

在线性代数、计算几何中，向量点积是一种十分重要的运算。给定两个 n 维向量 a=(a1,a2,...,an) 和 b=(b1,b2,...,bn)，求点积 a?b=a1b1+a2b2+...+anbn。

【输入】

第一行是一个整数 n(1 ≤ n ≤ 1000)；

第二行包含 n 个整数 a1,a2,...,an；

第三行包含 n 个整数 b1,b2,...,bn；

相邻整数之间用单个空格隔开。每个整数的绝对值都不超过 1000。

【输出】

一个整数，即两个向量的点积结果。

【输入样例】

3

1 4 6

2 1 5

【输出样例】

36

4.4-7

```
#include<iostream>//7. 向量点积计算
using namespace std;
int main()
{
	int a[1001],b[1001],i,n,s=0;
	cin>>n;
	for(i=1;i<=n;i++)		cin>>a[i];
	for(i=1;i<=n;i++)		cin>>b[i];
	for(i=1;i<=n;i++)
	{
		s+=a[i]*b[i];
	}
	cout<<s;
	return 0;
}
```

```
3
1 4 6
2 1 5
36
```

假设有 N 盏灯（N 为不大于 5000 的正整数），从 1 到 N 按顺序依次编号，初始时全部处于开启状态；有 M 个人（M 为不大于 N 的正整数）也从 1 到 M 依次编号。

第一个人（1 号）将灯全部关闭，第二个人（2 号）将编号为 2 的倍数的灯打开，第三个人（3 号）将编号为 3 的倍数的灯做相反处理（即将打开的灯关闭，将关闭的灯打开）。依照编号递增顺序，以后的人都和 3 号一样，将凡是自己编号倍数的灯做相反处理。

请问：当第 M 个人操作之后，哪几盏灯是关闭的，按从小到大输出其编号，其间用逗号间隔。

4.4-8// 选自《一本通》

```
#include<iostream>//8. 开关灯
#include<cstring>
using namespace std;
bool a[5001];// 用 int 也可以
int main()
{
	int i,j,n,m,z=1;
	cin>>n>>m;//n 盏灯，m 人操作
	memset(a,0,sizeof(a));// 初始化为 0，灯开启状态
	for(i=1;i<=m;i++)//m 人依次操作
	{
		for(j=1;j<=n;j++)//n 盏灯
		{
			if(j%i==0) // 如果是 i 的倍数，做相反处理
				a[j]=!a[j];
		}
	}

	for(i=1;i<=n;i++)
	{
		if(a[i])// 如果 a[i]==1，灯关闭状态
		{
			cout<<i<<" ";
		}
	}
	return 0;
}
```

```
10 10

1 4 9
```

```
30 7
1 4 8 10 11 12 13 14
15 16 17 19 21 23 24
29 30
```

【题目描述】

在一个序列（下标从 1 开始）中查找一个给定的值，输出第一次出现的位置。

【输入】

第一行包含一个正整数 n，表示序列中元素个数。1 <=n<= 10000。

第二行包含 n 个整数，依次给出序列的每个元素，相邻两个整数之间用单个空格隔开。元素的绝对值不超过 10000。

第三行包含一个整数 x，为需要查找的特定值。x 的绝对值不超过 10000。

【输出】

若序列中存在 x，输出 x 第一次出现的下标；否则输出 -1。

【输入样例】

5

2 3 6 7 3

3

【输出样例】

2

4.4-9

```
5
2 3 6 7 3
3

2
```

```
#include<iostream>//9. 查找特定的值
using namespace std;
int a[10001];
int main()
{
	int i,n,x,k=0;
	cin>>n;
	for(i=1;i<=n;i++) cin>>a[i];
	cin>>x;
	for(i=1;i<=n;i++)
	{
		if(a[i]==x)
		{
			cout<<i;
			break;
		}
	}
	return 0;
}
```

4.4-9

```
#include<iostream>//9. 查找特定的值，仅供参考
using namespace std;
int a[10001];
int main()
{
	int i,n,x,k;
	cin>>n;
	for(i=1;i<=n;i++)
		cin>>a[i];
	cin>>x;
	i=1; //从下标 1 开始查找
	k=0;
	while(k<1)//如果找到一个就退出循环
	{
		if(a[i]==x)
		{
			cout<<i;
			k=1;//标记已经找到一个，可以退出了
		}
		i++;//如果没有找到，继续查找
	}
	return 0;
}
```

【题目描述】

津津上初中了。妈妈认为津津应该更加用功学习，所以津津除了上学之外，还要参加妈妈为她报名的各科复习班。另外每周妈妈还会送她去学习朗诵、舞蹈和钢琴。但是津津如果一天上课超过八个小时就会不高兴，而且上得越久就会越不高兴。假设津津不会因为其它事不高兴，并且她的不高兴不会持续到第二天。请你帮忙检查一下津津下周的日程安排，看看下周她会不会不高兴；如果会的话，哪天最不高兴。

【输入】

包括七行数据，分别表示周一到周日的日程安排。每行包括两个小于 10 的非负整数，用空格隔开，分别表示津津在学校上课的时间和妈妈安排她上课的时间。

【输出】

包括一行，这一行只包含一个数字。如果不会不高兴则输出 0，如果会则输出最不高兴的是周几（用 1，2，3，4，5，6，7 分别表示周一，周二，周三，周四，周五，周六，周日）。如果有两天或两天以上不高兴的程度相当，则输出时间最靠前的一天。

【输入样例】

5 3

6 2

7 2

5 3

5 4

0 4

0 6

【输出样例】

3

4.4-10// 选自 noip2004 普及组

```
#include<iostream>//10. 不高兴的津津
using namespace std;
int main()
{
	int x,y,c[8],i,max=0;
	for(i=1;i<=7;i++)
	{
		cin>>x>>y;
		c[i]=x+y;// 统计每天学习时间
		if(c[i]>max&&c[i]>8)// 如果学习时间大于 8，查找出最大数字
			max=c[i];
	}
	for(i=1;i<=7;i++)
	{
		if(c[i]==max)// 如果存在超过 8 的数字
		{
			cout<<i;// 输出之后马上跳出循环
			break;
		}
		if(max==0)
		{
			cout<<0;
			break;
		}

	}
	return 0;
}
```

```
1 1
2 1
1 2
2 2
1 1
1 1
1 1
0
```

```
5 3
6 2
7 2
5 3
5 4
0 4
0 6
3
```

【题目描述】

输出一个整数序列中最大的数和最小的数的差。

【输入】

第一行为 M，表示整数个数，整数个数不会大于 10000；

第二行为 M 个整数，以空格隔开，每个整数的绝对值不会大于 10000。

【输出】

输出 M 个数中最大值和最小值的差。

【输入样例】

5

2 5 7 4 2

【输出样例】

5

4.4-11

```
5
2 5 7 4 2
5
```

```
#include<iostream>//11. 最大值和最小值的差
using namespace std;
int main()
{
	int i,a[10001],m,max,min;//max 最大数，min 最小数
	cin>>m;
	min=INT_MAX;// 预设最小数
	max=INT_MIN;// 预设最大数

	for(i=1;i<=m;i++)
	{
		cin>>a[i];
		if(a[i]<min)	min=a[i];
		if(a[i]>max)	max=a[i];
	}
	cout<<max-min;
	return 0;
}
```

【题目描述】输出一个整数数列中不与最大数相同的数字之和。

【输入】

输入分为两行：

第一行为N(N为接下来数的个数，N <= 100)；

第二行N个整数，数与数之间以一个空格分开，每个整数的范围是-1000,000到1000,000。

【输出】输出为N个数中除去最大数其余数字之和。

【输入样例】

3

1 2 3

【输出样例】

3

4.4-12

```
#include<iostream>//12. 不与最大数相同的数字之和
using namespace std;
int main()
{
	int a[1001],n,s=0,i,max;
	max=-10000;
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i];
		if(a[i]>max)
			max=a[i];
	}
	for(i=1;i<=n;i++)
	{
		if(a[i]!=max)
			s+=a[i];
	}
	cout<<s;
}
```

```
3
1 2 3
3
```

【题目描述】

医院采样了某临床病例治疗期间的白细胞数量样本 n 份，用于分析某种新抗生素对该病例的治疗效果。为了降低分析误差，要先从这 n 份样本中去除一个数值最大的样本和一个数值最小的样本，然后将剩余 n-2 个有效样本的平均值作为分析指标。同时，为了观察该抗生素的疗效是否稳定，还要给出该平均值的误差，即所有有效样本（即不包括已扣除的两个样本）与该平均值之差的绝对值的最大值。

现在请你编写程序，根据提供的 n 个样本值，计算出该病例的平均白细胞数量和对应的误差。

【输入】

输入的第一行是一个正整数 n（2<n ≤ 300），表明共有 n 个样本。

以下共有 n 行，每行为一个浮点数，为对应的白细胞数量，其单位为 109/L。数与数之间以一个空格分开。

【输出】

输出为两个浮点数，中间以一个空格分开。分别为平均白细胞数量和对应的误差，单位也是 109/L。计算结果需保留到小数点后 2 位。

【输入样例】

5

12.0

13.0

11.0

9.0

10.0

【输出样例】

11.00 1.00

4.4-13

```
#include<iostream>//13. 白细胞计数
#include<cstdio>
#include<cmath>
using namespace std;
double a[1001],b[1001];
int main()
{
	int i,k,v;
	double w,max=-1000,min=10001,s=0,n;// 预设最大数最小数
	cin>>n;
	for(i=1;i<=n;i++)// 输入的同时，查找最大数字，最小数字
	{
		cin>>a[i];
		s+=a[i];
		if(a[i]>max) {	max=a[i];k=i;	}
		if(a[i]<min) {	min=a[i];v=i;	}
	}
	w=(s-a[v]-a[k])/(n-2.0);// 去掉最大数字、最小数字之后，求平均值
	printf("%.2f\n",w);

	for(i=1;i<=n;i++)
	{
		if(i!=k&&i!=v)	b[i]=abs(w-a[i]);
		//abs 是求数据绝对值函数，b 数组存储 a 数组的数字与平均值误差
		else		b[i]=0;// 最大数字、最小数字与平均值的误差都设置为 0
	}
	max=-10001;// 预设最大数字，然后开始找最大数
	for(i=1;i<=n;i++)
	{
		if(b[i]>max)		max=b[i];
	}
	printf("%.2f",max);
	return 0;
}
```

【题目描述】给定一个非负整数数组，统计里面每一个数的出现次数。我们只统计到数组里最大的数。

假设 Fmax（Fmax<10000）是数组里最大的数，那么我们只统计 {0,1,2.....Fmax} 里每个数出现的次数。

【输入】第一行 n 是数组的大小。1 ≤ n ≤ 10000。紧接着一行是数组的 n 个元素。

【输出】按顺序输出每个数的出现次数，一行一个数。如果没有出现过，则输出 0。对于例子中的数组，最大的数是 3，因此我们只统计 {0,1,2,3} 的出现频数。

【输入样例】

5

1 1 2 3 1

【输出样例】

0 3 1 1

4.4-14

```
#include<iostream>//14. 直方图
#include<cstring>
using namespace std;
int a[10001],i,n,x,maxn=0;
int main()
{
	memset(a,0,sizeof(0));// 数组 a，用来统计数字出现频率，初始化为 0 次
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>x;
		a[x]++;
		if(x>maxn)
			maxn=x;// 查找最大数字
	}

	for(i=0;i<=maxn;i++)
	{
		cout<<a[i]<<" ";
	}
	return 0;
}
```

【题目描述】

已知一个已经从小到大排序的数组，这个数组的一个平台（Plateau）就是连续的一串值相同的元素，并且这一串元素不能再延伸。例如，在 1，2，2，3，3，3，4，5，5，6 中 1，2-2，3-3-3，4，5-5，6 都是平台。试编写一个程序，接收一个数组，把这个数组最长的平台找出来。在上面的例子中 3-3-3 就是最长的平台。

【输入】

第一行有一个整数 n，为数组元素的个数。第二行有 n 个整数，整数之间以一个空格分开。

【输出】

输出最长平台的长度。

【输入样例】

10

1 2 2 3 3 3 4 5 5 6

【输出样例】

3

4.4-15

```
#include<iostream>//15. 最长平台
using namespace std;
int a[100001],n,i,k=1,maxn=0;
int main()
{
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i];
		if(a[i]==a[i-1])
			k++;
		else
			k=1;
		if(k>maxn)
			maxn=k;// 寻找最长平台
	}
	cout<<maxn;
	return 0;
}
```

```
4.4-16// 选自《一本通》
#include<iostream>//16. 整数去重
#include<cstring>
using namespace std;
int a[20001];
bool s[20001];
int main()
{
	int i,j,n;
	cin>>n;
	memset(s,true,sizeof(s));
	// 初始化为真，表示没有重复
	for(i=1;i<=n;i++)
		cin>>a[i];
	for(i=1;i<=n;i++)
	{
		if(s[i])
		{
			for(j=i+1;j<=n;j++)
			{
				if(a[i]==a[j])
				{
					s[j]=false;// 发现相等的，设置为假
				}
			}
		}
	}
	for(i=1;i<=n;i++)
	{
		if(s[i]==true)
			cout<<a[i]<<" ";
	}
	return 0;
}
```

【题目描述】给定含有 n 个整数的序列，要求对这个序列进行去重操作。所谓去重，是指对这个序列中每个重复出现的数，只保留该数第一次出现的位置，删除其余位置。

【输入】输入包含两行：

第一行包含一个正整数 n（1 ≤ n ≤ 20000），表示第二行序列中数字的个数；

第二行包含 n 个整数，整数之间以一个空格分开。每个整数大于等于 10、小于等于 5000。

【输出】输出只有一行，按照输入的顺序输出其中不重复的数字，整数之间用一个空格分开。

【输入样例】

5

10 12 93 12 75

【输出样例】

10 12 93 75

mespac【题目描述】

为了准备一个独特的颁奖典礼，组织者在会场的一片矩形区域（可看做是平面直角坐标系的第一象限）铺上一些矩形地毯。一共有 n 张地毯，编号从 1 到 n。现在将这些地毯按照编号从小到大的顺序平行于坐标轴先后铺设，后铺的地毯覆盖在前面已经铺好的地毯之上。地毯铺设完成后，组织者想知道覆盖地面某个点的最上面的那张地毯的编号。注意：在矩形地毯边界和四个顶点上的点也算被地毯覆盖。

输入输出样例 1 说明：如下图，1 号地毯用实线表示，2 号地毯用虚线表示，3 号用双实线表示，覆盖点（2，2）的最上面一张地毯是 3 号地毯。

输入输出样例 2 说明：如下图，1 号地毯用实线表示，2 号地毯用虚线表示，3 号用双实线表示，覆盖点（4，5）的最上面没有一张地毯。

【输入】

第一行，一个整数 n，表示总共有 n 张地毯。

接下来的 n 行中，第 i+1 行表示编号 i 的地毯的信息，包含四个正整数 a，b，g，k，每两个整数之间用一个空格隔开，分别表示铺设地毯的左下角的坐标（a，b）以及地毯在 x 轴和 y 轴方向的长度。

第 n+2 行包含两个正整数 x 和 y，表示所求的地面的点的坐标（x，y）。

【输出】

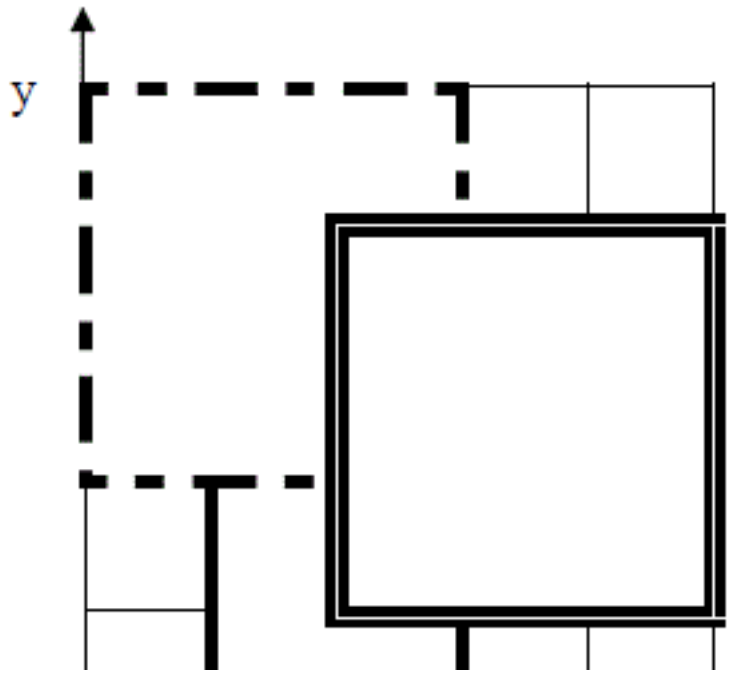
输出共 1 行，一个整数，表示所求的地毯的编号；若此处没有被地毯覆盖则输出 -1。

【输入样例】

```
3
1 0 2 3
0 2 3 3
2 1 3 3
2 2
```

【输出样例】

```
3
```



样例输入 #2：

```
3
1 0 2 3
0 2 3 3
2 1 3 3
4 5
```

样例输出 #2：

```
-1
```

【数据范围】全部数据，1 ≤ n ≤ 10000。

4.4-17

```
#include<iostream>//17. 铺地毯
using namespace std;
int main()
{
	int i,n,x,y,a[10001],b[10001],g[10001],k[10001];
	cin>>n;
	for(i=1;i<=n;i++)
	{
		cin>>a[i]>>b[i]>>g[i]>>k[i];
		// 地毯左下角坐标（a,b），地毯在 x 轴和 y 轴方向的长度
	}
	cin>>x>>y;

	for(i=n;i>=1;i--)// 找最上面覆盖的地毯，所以从上往下找
	{
		if(x>=a[i]&&x<=a[i]+g[i]&&y>=b[i]&&y<=b[i]+k[i])
		{
			cout<<i;
			return 0;// 找到第一张地毯后立即退出程序
		}
	}
	cout<<-1;
// 如果没找到，输出 -1 （前面的 for 循环运行结束没找到地毯，自动向下输出 -1）
	return 0;
}
```

《一本通训练指导》二维数组

4.7-1

```
#include<iostream>// 交换第一行和第五行数字
using namespace std;
int main()
{
	int n,m,a[6][6],t,i,j;
	for(i=1;i<=5;i++)// 输入数组
	{
		for(j=1;j<=5;j++)
		{
			cin>>a[i][j];
		}
	}
	//////////////////////////////////////////
	cin>>m>>n;// 交换行
	for(t=1;t<=5;t++)
	{
		swap(a[m][t],a[n][t]);
	}
	//////////////////////////////////////////
	for(i=1;i<=5;i++)// 输出交换后的二维数组
	{
		for(j=1;j<=5;j++)
		{
			cout<<a[i][j]<<" ";
		}
		cout<<endl;
	}
	return 0;
}
```

```
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
51 52 53 54 55
1 5
51 52 53 54 55
21 22 23 24 25
31 32 33 34 35
41 42 43 44 45
11 12 13 14 15
```

4.7-2

```
4 2 3
2 1   2 2   2 3   2 4

1 3   2 3   3 3   4 3

1 2   2 3   3 4

4 1   3 2   2 3   1 4
```

```
#include<iostream>// 位置输出
using namespace std;
int main()
{
	int n,i,j,t,s,x,y;
	int a[11][11];
	cin>>n>>x>>y;

	for(i=1;i<=n;i++)// 同一行格子位置输出
		cout<<x<<" "<<i<<"   ";
	cout<<endl; cout<<endl;

	for(i=1;i<=n;i++)	cout<<i<<" "<<y<<"   ";// 同一列格子位置输出
	cout<<endl; cout<<endl;

	for(i=1;i<=n;i++)// 从左上到右下输出同一对角线格子位置
	{
		for(j=1;j<=n;j++)
		{
			if(x-y==i-j)	cout<<i<<" "<<j<<"   ";
		}
	}
	cout<<endl;cout<<endl;

	for(i=n;i>=1;i--)// 从左下到右上输出同一对角线格子位置
	{
		for(j=n;j>=1;j--)
		{
			if(x+y==i+j)	cout<<i<<" "<<j<<"   ";
		}
	}
	return 0;
}
```

4.7-3

```
3 3
3 4 1
3 7 1
2 0 1
15
```

```
#include<iostream>// 计算矩阵周边元素之和
using namespace std;
int main()
{
	int a[101][101],m,n,i,j,s=0;
	cin>>m>>n;
	for(i=1;i<=m;i++)
	{
		for(j=1;j<=n;j++)
		{
			cin>>a[i][j];
			if(i==1||j==1||i==m||j==n) s=s+a[i][j];
		}
	}
	cout<<s;
	return 0;
}
```

4.7-4

```
11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25
4 1 8
```

```
#include<iostream>// 寻找鞍点
using namespace std;// 鞍点是行中最大，列中最小的数
int a[6][6];
int main(){
	int i,j,x=0,y,z;
	bool q;// 判断是否找到鞍点
	for(i=1;i<=5;i++)// 输入矩阵
		for(j=1;j<=5;j++)		cin>>a[i][j];
	for(i=1;i<=5;i++)// 枚举每一行
	{
		q=true;
		x=0;
		for(j=1;j<=5;j++)// 枚举每一列
		{
			if(a[i][j]>x)// 找出每一行最大数字
			{
				x=a[i][j];
				y=j;
				z=i;
			}
			for(j=1;j<=5;j++)// 判断该行最大数字是否该列最小数
			{
				if(a[j][y]<x)// 如果找到更小的数字，寻找鞍点失败
					q=false;
			}
			if(q==true)// 如果没有找到更小的数字，那么寻找鞍点成功
			{	cout<<z<<" "<<y<<" "<<x<<endl;
				return 0;
			}
		}
	}
	cout<<"not found";// 没有鞍点
	return 0;
}
```

4.7-5

```
3 3
1 0 1
0 0 1
1 1 0

1 1 0
0 0 1
0 0 1

44.44
44.4444
```

```
#include<cstdio>
#include<iostream>// 图像相似度
using namespace std;
int main()
{
	int a[101][101],b[101][101];
	int i,j,m,n;
	double s=0;// 图像相似暂时设置为 0
	cin>>m>>n;//m,n 是图像的行和列

	for(i=1;i<=m;i++)// 读入第一个矩阵
	{
		for(j=1;j<=n;j++)
		{
			cin>>a[i][j];
		}
	}

	for(i=1;i<=m;i++)// 读入第二个矩阵，同时对比相同位置上的元素
	{
		for(j=1;j<=n;j++)
		{
			cin>>b[i][j];
			if(a[i][j]==b[i][j])
				s++;
		}
	}
	printf("%.2f",s/(m*n)*100);// 准确到小数点后两位
	cout<<endl<<s/(m*n)*100;
	return 0;
}
```

4.7-6

```
#include<iostream>// 矩阵加法
using namespace std;
int main()
{
	int i,j,n,m;
	int a[101][101],b[101][101];
	cin>>n>>m;
	for(i=1;i<=n;i++)// 输入第一个矩阵
	{
		for(j=1;j<=m;j++)
		{
			cin>>a[i][j];
		}
	}
	for(i=1;i<=n;i++)// 输入第二个矩阵
	{
		for(j=1;j<=m;j++)
		{
			cin>>b[i][j];
		}
	}

	for(i=1;i<=n;i++)// 输出矩阵之和
	{
		for(j=1;j<=m;j++)
		{
			cout<<a[i][j]+b[i][j]<<" ";
		}
		cout<<endl;
	}
	return 0;
}
```

```
3 3

1 2 3
1 2 3
1 2 3

1 2 3
4 5 6
7 8 9

2 4 6
5 7 9
8 10 12
```

4.7-7

```
3 2 3

1 1
1 1
1 1

1 1 1
1 1 1

2 2 2
2 2 2
2 2 2
```

```
#include<cstdio>// 矩阵乘法
#include<iostream>
using namespace std;
int main()
{
	int i,j,t,n,m,k;
	int a[100][100],b[100][100],c[100][100];
	cin>>n>>m>>k;
	for(i=1;i<=n;i++)// 输入第一个矩阵
	{
		for(j=1;j<=m;j++)
			cin>>a[i][j];
	}
	for(i=1;i<=m;i++)// 输入第二个矩阵
	{
		for(j=1;j<=k;j++)
			cin>>b[i][j];
	}
	for(i=1;i<=n;i++)// 开始累加 c[i][j] 的值
	{
		for(j=1;j<=k;j++)
		{
			for(t=1;t<=m;t++)
				c[i][j]+=a[i][t]*b[t][j];// 累加 c[i][j] 的值
		}
	}
	for(i=1;i<=n;i++)// 输出乘法
	{
		for(j=1;j<=k;j++)
			cout<<c[i][j]<<" ";
		cout<<endl;
	}
	return 0;
}
```

4.7-8

```
#include<iostream>// 矩阵转置
using namespace std;
int main()
{
	int a[100][100];
	int m,n,i,j;
	cin>>n>>m;
	for(i=1;i<=n;i++)// 读入矩阵
	{
		for(j=1;j<=m;j++)
		{
			cin>>a[i][j];
		}
	}

	for(i=1;i<=m;i++)
	{
		for(j=1;j<=n;j++)
		{
			cout<<a[j][i]<<" ";
		}
		cout<<endl;
	}
	return 0;
}
```

```
3 3

1 2 3
4 5 6
7 8 9

1 4 7
2 5 8
3 6 9
```

4.7-9

```
3 3

1 2 3
4 5 6
7 8 9

7 4 1
8 5 2
9 6 3
```

```
#include<iostream>// 图像旋转 90 度
using namespace std;
int main()
{
	int a[100][100];// 数组不要设置太大，难以运行
	int m,n,i,j;
	cin>>n>>m;
	for(i=1;i<=n;i++)// 读入矩阵
	{
		for(j=1;j<=m;j++)
		{
			cin>>a[i][j];
		}
	}

	for(i=1;i<=m;i++)// 从第一列到最后一列，从最后一行到第一行排列
	{
		for(j=n;j>=1;j--)
		{
			cout<<a[j][i]<<" ";
		}
		cout<<endl;
	}
	return 0;
}
```

4.7-10

```
4 5
100 0 100 0 50
50 100 200 0 0
50 50 100 100 200
100 100 50 50 100


100 0 100 0 50
50 80 100 60 0
50 80 100 90 200
100 100 50 50 100
```

```
#include<iostream>// 图像模糊处理
#include<cmath>
using namespace std;
int main()
{
	int a[100][100];// 数值设置太大难以运行
	double b[100][100];// 数值设置太大难以运行
	int n,m,i,j;
	cin>>n>>m;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=m;j++)
		{
			cin>>a[i][j];
			b[i][j]=a[i][j];// 需要保存原图和模糊后的图像
		}
	}
	for(i=2;i<=n-1;i++)
	{
		for(j=2;j<=m-1;j++)
		{
	b[i][j]=round((a[i-1][j]+a[i][j-1]+a[i+1][j]+a[i][j+1]+a[i][j])/5.0);
//round 四舍五入，各像素点新灰度值为该像素点及其上下左右四个像素点原灰度值平均值
		}
	}
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=m;j++)
			cout<<b[i][j]<<" ";
		cout<<endl;
	}
	return 0;
}
```