CCF 历年真题（C++实现）

目录

2015 年 9 月

数列分段

试题编号：	201509-1
试题名称：	数列分段
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 给定一个整数数列，数列中连续相同的最长整数序列算成一段，问数列中共有多少段？ **输入格式** 输入的第一行包含一个整数 n，表示数列中整数的个数。 第二行包含 n 个整数 a_1，a_2，…，a_n，表示给定的数列，相邻的整数之间用一个空格分隔。 **输出格式** 输出一个整数，表示给定的数列有多个段。 **样例输入** 8 8 8 8 0 12 12 8 0 **样例输出** 5 **样例说明** 8 8 8 是第一段，0 是第二段，12 12 是第三段，倒数第二个整数 8 是第四段，最后一个 0 是第五段。 **评测用例规模与约定** 1 ≤ n ≤ 1000，0 ≤ a_i ≤ 1000。

答案参考（仅代表个人观点）

```
# include<iostream>
using namespace std;
int main(){
	int a[1000],b,c=0,d;
	cin>>b;
	for(int i=0;i<b;i++){
		cin>>a[i];
	}
	d=a[0];
```

```
        c=1;
        for(int i=0;i<b;i++){
            if(a[i]!=d){
                c++;
                d=a[i];
            }
        }
        cout<<c;
        return 0;
}
```

日期计算

问题描述

试题编号：	201509-2
试题名称：	日期计算
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 给定一个年份 y 和一个整数 d，问这一年的第 d 天是几月几日？ 注意闰年的 2 月有 29 天。满足下面条件之一的是闰年： 1） 年份是 4 的整数倍，而且不是 100 的整数倍； 2） 年份是 400 的整数倍。 **输入格式** 输入的第一行包含一个整数 y，表示年份，年份在 1900 到 2015 之间（包含 1900 和 2015）。 输入的第二行包含一个整数 d，d 在 1 至 365 之间。 **输出格式** 输出两行，每行一个整数，分别表示答案的月份和日期。 **样例输入** 2015 80 **样例输出** 3 21 **样例输入**

	2000 40 **样例输出** 2 9

```
# include<iostream>
using namespace std;
int main(){
	int monthDay[]={31,28,31,30,31,30,31,31,30,31,30,31};
	int year;
	int day;
	int count;
	int month;
	cin>>year;
	cin>>day;
	if(((year%4==0)&&(year%100!=0))||(year%400==0)){
		monthDay[1]=29;
	}
	for(int i=0;i<12,count<day;i++){
		count+=monthDay[i];
		month=i;
	}
	count-=monthDay[month];
	cout<<month+1<<endl<<day-count;
	return 0;
}
```

模板生成工具

问题描述

试题编号：	201509-3
试题	模板生成系统

名称：	
时间限制：	1.0s
内存限制：	256.0MB

问题描述：	**问题描述**
	成成最近在搭建一个网站，其中一些页面的部分内容来自数据库中不同的数据记录，但是页面的基本结构是相同的。例如，对于展示用户信息的页面，当用户为 Tom 时，网页的源代码是
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`  <title>User Tom</title>`
	`  </head>`
	`  <body>`
	`  <h1>Tom</h1>`
	`  <p>Email: <a href="mailto:tom@example.com">tom@example.com</a><`
	`  </body>`
	`</html>`
	而当用户为 Jerry 时，网页的源代码是
	`<!DOCTYPE html>`
	`<html>`
	`  <head>`
	`  <title>User Jerry</title>`
	`  </head>`
	`  <body>`
	`  <h1>Jerry</h1>`
	`  <p>Email: <a href="mailto:jerry@example.com">jerry@example.com<`
	`  </body>`
	`</html>`

这样的例子在包含动态内容的网站中还有很多。为了简化生成网页的工作，成成觉得他需要引入一套模板生成系统。

模板是包含特殊标记的文本。成成用到的模板只包含一种特殊标记，格式为{{VAR}}，其中 VAR 是一个变量。该标记在模板生成时会被变量 VAR 的值所替代。例如，如果变量 name = "Tom"，则 {{ name }} 会生成 Tom。具体的规则如下：

- 变量名由大小写字母、数字和下划线（_）构成，且第一个字符不是数字，长度不超过 16 个字符。
- 变量名是大小写敏感的，Name 和 name 是两个不同的变量。
- 变量的值是字符串。
- 如果标记中的变量没有定义，则生成空串，相当于把标记从模板中删除。
- 模板不递归生成。也就是说，如果变量的值中包含形如 {{ VAR }} 的内容，不再做进一步的替换。

输入格式

输入的第一行包含两个整数 m, n，分别表示模板的行数和模板生成时给出的变量个数。

接下来 m 行，每行是一个字符串，表示模板。

接下来 n 行，每行表示一个变量和它的值，中间用一个空格分隔。值是字符串，用双引号（"）括起来，内容可包含除双引号以外的任意可打印 ASCII 字符（ASCII 码范围 32, 33, 35-126）。

输出格式

输出包含若干行，表示模板生成的结果。

样例输入

```
11 2
<!DOCTYPE html>
<html>
<head>
<title>User {{ name }}</title>
</head>
<body>
<h1>{{ name }}</h1>
<p>Email: <a
href="mailto:{{ email }}">{{ email }}</a></p>
<p>Address: {{ address }}</p>
</body>
</html>
name "David Beckham"
email "david@beckham.com"
```

样例输出

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>User David Beckham</title>
</head>
<body>
<h1>David Beckham</h1>
<p>Email: <a
href="mailto:david@beckham.com">david@beckham.com</a
></p>
<p>Address: </p>
</body>
</html>
```

评测用例规模与约定

0 ≤ m ≤ 100

0 ≤ n ≤ 100

输入的模板每行长度不超过 80 个字符（不包含换行符）。

输入保证模板中所有以 {{ 开始的子串都是合法的标记，开始是两个左大括号和一个空格，然后是变量名，结尾是一个空格和两个右大括号。

输入中所有变量的值字符串长度不超过 100 个字符（不包括双引号）。

保证输入的所有变量的名字各不相同。

```
# include<iostream>
# include<vector>
# include<string>
# include<map>
using namespace std;
int main(){
    int lineNum=0,varNum=0;
    vector<string> content;
    string line;
    map<string,string> varMap;
    cin>>lineNum>>varNum;
    cin.ignore();
    for(int i=0;i<lineNum;i++){
        getline(cin,line);
        content.push_back(line);
    }
    for(int i=0;i<varNum;i++){
        getline(cin,line);
        int pos = line.find(" ");
        varMap.insert(map<string,string>::value_type(line.substr(0,pos),line.substr(pos)));
    }
    for(int i=0;i<lineNum;i++){
        int pos =0,pos1,pos2;
```

```
            do{
                pos1=content[i].find("{{",pos);
                pos2 = content[i].find("}}",pos1);
                    if(pos1>=0 && pos2>=0){
                            string var = content[i].substr(pos1+3,pos2-pos1-4);
                            if(varMap.count(var)){
                                    string result = varMap[var].substr(2,varMap[var].length()-3);
                                    content[i].replace(pos1,var.length()+6,result);
                            }else{
                                    content[i].replace(pos1,var.length()+6,"");
                            }
                            pos=pos1+var.length();
                    }else{
                            pos = content[i].length();
                    }
            }while(pos<content[i].length());
            cout<<content[i]<<endl;
        }
        return 0;
}
```

2015 年 3 月

图像旋转

问题描述

试题编号：	201503-1
试题名称：	图像旋转
时间限制：	5.0s
内存限制：	256.0MB
问题描述：	**问题描述** 旋转是图像处理的基本操作，在这个问题中，你需要将一个图像逆时

针旋转 90 度。

计算机中的图像表示可以用一个矩阵来表示，为了旋转一个图像，只需要将对应的矩阵旋转即可。

输入格式

输入的第一行包含两个整数 n，m，分别表示图像矩阵的行数和列数。

接下来 n 行每行包含 m 个整数，表示输入的图像。

输出格式

输出 m 行，每行包含 n 个整数，表示原始矩阵逆时针旋转 90 度后的矩阵。

样例输入

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

样例输出

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

评测用例规模与约定

1 ≤ n，m ≤ 1,000，矩阵中的数都是不超过 1000 的非负整数。

```
# include<iostream>
#include<vector>
using namespace std;
int main(){
	int m,n;
	cin>>m>>n;
	typedef vector<int> valVec;
	vector<valVec> arr(m,vector<int>(n,0));
	for(int i=0;i<m;i++){
		for(int j=0;j<n;j++){
			int value;
			cin>>value;
			arr[i][j]=value;
		}
	}
	for(int i=n-1;i>=0;i--){
		for(int j=0;j<m;j++){
			cout<<arr[j][i]<<" ";
		}
		cout<<endl;
	}
```

```
    //system("pause");
    return 0;
}
```

数字排序

问题描述

试题编号：	201503-2
试题名称：	数字排序
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 给定 n 个整数，请统计出每个整数出现的次数，按出现次数从多到少的顺序输出。 **输入格式** 输入的第一行包含一个整数 n，表示给定数字的个数。 第二行包含 n 个整数，相邻的整数之间用一个空格分隔，表示所给定的整数。 **输出格式** 输出多行，每行包含两个整数，分别表示一个给定的整数和它出现的次数。按出现次数递减的顺序输出。如果两个整数出现的次数一样多，则先输出值较小的，然后输出值较大的。 **样例输入** 12 5 2 3 3 1 3 4 2 5 2 3 5 **样例输出** 3 4 2 3 5 3 1 1 4 1 **评测用例规模与约定** 1 ≤ n ≤ 1000，给出的数都是不超过 1000 的非负整数。

```
#include<iostream>
#include<map>
#include<vector>
#include<algorithm>
using namespace std;
typedef pair<int,int> PAIR;
typedef map<int,int>::value_type valType;
struct CmpByValue{
        bool operator()(const PAIR& lhs, const PAIR& rhs) {
                bool result;
                if(lhs.second == rhs.second){
                        result = lhs.first < rhs.first;
                }else {
                   result= lhs.second > rhs.second;
                }
                return result;
         }
};
int main(){
       int n;
       cin>>n;
       int arr[1000] = {0};
       map<int,int> countMap;
       for(int i=0;i<n;i++){
              int value;
              cin>>value;
              arr[i]=value;
              ++countMap[value];
       }
       vector<PAIR> valueVector(countMap.begin(),countMap.end());
       sort(valueVector.begin(),valueVector.end(),CmpByValue());
        for (int i = 0; i != valueVector.size(); ++i) {
                cout<<valueVector[i].first<<" "<<valueVector[i].second<< endl;
        }
              return 0;
}
```

节日

问题描述

试题编号：	201503-3
试题名称：	节日
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 有一类节日的日期并不是固定的，而是以“a 月的第 b 个星期 c”的形式定下来的，比如说母亲节就定为每年的五月的第二个星期日。 现在，给你 a，b，c 和 y_1，y_2(1850 ≤ y_1，y_2 ≤ 2050)，希望你输出从公元 y_1 年到公元 y_2 年间的每年的 a 月的第 b 个星期 c 的日期。 提示：关于闰年的规则：年份是 400 的整数倍时是闰年，否则年份是 4 的倍数并且不是 100 的倍数时是闰年，其他年份都不是闰年。例如 1900 年就不是闰年，而 2000 年是闰年。 为了方便你推算，已知 1850 年 1 月 1 日是星期二。 **输入格式** 输入包含恰好一行，有五个整数 a，b，c，y_1，y_2。其中 c=1，2，……，6，7 分别表示星期一、二、……、六、日。 **输出格式** 对于 y_1 和 y_2 之间的每一个年份，包括 y_1 和 y_2，按照年份从小到大的顺序输出一行。 如果该年的 a 月第 b 个星期 c 确实存在，则以"yyyy/mm/dd"的格式输出，即输出四位数的年份，两位数的月份，两位数的日期，中间用斜杠“/”分隔，位数不足时前补零。 如果该年的 a 月第 b 个星期 c 并不存在，则输出"none"（不包含双引号）。 **样例输入** 5 2 7 2014 2015 **样例输出** 2014/05/11 2015/05/10 **评测用例规模与约定** 所有评测用例都满足：1 ≤ a ≤ 12，1 ≤ b ≤ 5，1 ≤ c ≤ 7，1850 ≤ y_1，y_2 ≤ 2050。

```
#include<iostream>
using namespace std;
  int IsLeapYear(int y )
{
```

```
    if((y%4==0 && y%100 !=0)||(y%400==0))
        return 1;
    else
        return 0;
}
int getTotalDay(int y,int m,int d)
{
    int Totalday=0;
    int MonthDays[12]={0,31,28,31,30,31,30,31,31,30,31,30};
    for(int i=1850;i<y;i++)
    {
        if(IsLeapYear(i))
            Totalday=Totalday+366;
        else
            Totalday=Totalday+365;
    }
    for(int i=0;i<m;i++)
    {
         Totalday=Totalday+MonthDays[i];
    }
    if(m>2)
        Totalday=Totalday+IsLeapYear(y);
    return Totalday+d;

}
int main (){
    int a,b,c,y1,y2;
    cin>>a>>b>>c>>y1>>y2;
    int MonthDays[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    for(int i=y1;i<=y2;i++){
        int totalDay = getTotalDay(i,a,1);
        int w = (totalDay+1)%7;
        if(w==0){
            w=7;
        }
        int day = 1+7*(b-1);
        if(w<c){
            day = day+c-w;
        }else if(w>c){
            day = day+7-w+c;
        }

        if(IsLeapYear(i)){
            MonthDays[2]=29;
```

```
            }else{
                MonthDays[2]=28;
            }
            if(day>MonthDays[a]){
                cout<<"none"<<endl;
            }else{
                cout<<i<<"/";
                if(a<10){
                    cout<<"0"<<a;
                }else{
                    cout<<a;
                }
                if(day<10){
                    cout<<"/"<<"0"<<day<<endl;
                }else{
                  cout<<"/"<<day<<endl;
                }
            }
        }
        return 0;
}
```

2014 年 12 月

门禁系统

问题描述

试题编号：	201412-1
试题名称：	门禁系统
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 涛涛最近要负责图书馆的管理工作，需要记录下每天读者的到访情况。每位读者有一个编号，每条记录用读者的编号来表示。给出读者的来访记

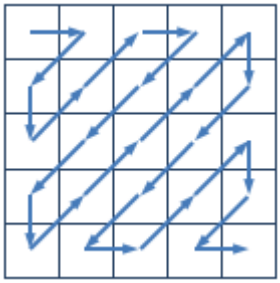
	录，请问每一条记录中的读者是第几次出现。 **输入格式** 输入的第一行包含一个整数 n，表示涛涛的记录条数。 第二行包含 n 个整数，依次表示涛涛的记录中每位读者的编号。 **输出格式** 输出一行，包含 n 个整数，由空格分隔，依次表示每条记录中的读者编号是第几次出现。 **样例输入** 5 1 2 1 1 3 **样例输出** 1 1 2 3 1 **评测用例规模与约定** 1≤n≤1,000，读者的编号为不超过 n 的正整数。

```
#include<iostream>
#include<map>
using namespace std;
int main(){
	int n;
	int record[1000]={0};
	map<int,int> analMap;
	cin>>n;
	for(int i=0;i<n;i++){
		cin>>record[i];
		++analMap[record[i]];
		record[i]=analMap[record[i]];
	}
	for(int i=0;i<n;i++){
		cout<<record[i]<<" ";
	}
	return 0;
}
```

Z 字形扫描

问题描述

试题编号：	201412-2

试题名称：	Z 字形扫描
时间限制：	2.0s
内存限制：	256.0MB
问题描述：	**问题描述** 在图像编码的算法中，需要将一个给定的方形矩阵进行 Z 字形扫描(Zigzag Scan)。给定一个 n×n 的矩阵，Z 字形扫描的过程如下图所示：  对于下面的 4×4 的矩阵， 1 5 3 9 3 7 5 6 9 4 6 4 7 3 1 3 对其进行 Z 字形扫描后得到长度为 16 的序列： 1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3 请实现一个 Z 字形扫描的程序，给定一个 n×n 的矩阵，输出对这个矩阵进行 Z 字形扫描的结果。 **输入格式** 输入的第一行包含一个整数 n，表示矩阵的大小。 输入的第二行到第 n+1 行每行包含 n 个正整数，由空格分隔，表示给定的矩阵。 **输出格式** 输出一行，包含 n×n 个整数，由空格分隔，表示输入的矩阵经过 Z 字形扫描后的结果。 **样例输入** 4 1 5 3 9 3 7 5 6 9 4 6 4 7 3 1 3 **样例输出** 1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3 **评测用例规模与约定**

	1≤n≤500，矩阵元素为不超过 1000 的正整数。

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
	int n;
	cin>>n;
	typedef vector<int> typeVec;
	vector<typeVec> arr(n,vector<int>(n,0));
	for(int i=0;i<n;i++){
		for(int j=0;j<n;j++){
			int value;
			cin>>value;
			arr[i][j]=value;
		}
	}
	for(int i=1;i<=n;i++){
		int temp=i-1;
			for(int j=0;j<i;j++,temp--){
				if(i%2==0){
				cout<<arr[j][temp]<<" ";
				}else{
				cout<<arr[temp][j]<<" ";
			}
		}
	}
	for(int i=n-1;i>0;i--){
		int temp=n-1;
			for(int j=n-i;j<n;j++,temp--){
				if(i%2==0){
					cout<<arr[j][temp]<<" ";
				}else{
					cout<<arr[temp][j]<<" ";
				}
		}
	}
	return 0;
}
```

2014 年 9 月

相邻数对

问题描述

试题编号：	201409-1
试题名称：	相邻数对
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 给定 n 个不同的整数，问这些数中有多少对整数，它们的值正好相差 1。 **输入格式** 输入的第一行包含一个整数 n，表示给定整数的个数。 第二行包含所给定的 n 个整数。 **输出格式** 输出一个整数，表示值正好相差 1 的数对的个数。 **样例输入** 6 10 2 6 3 7 8 **样例输出** 3 **样例说明** 值正好相差 1 的数对包括(2, 3), (6, 7), (7, 8)。 **评测用例规模与约定** 1<=n<=1000，给定的整数为不超过 10000 的非负整数。

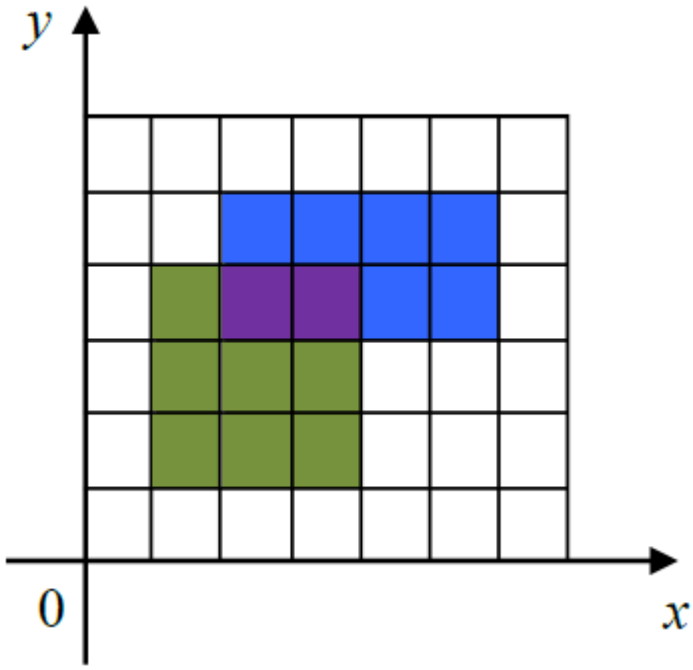
```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int main(){
	int n;
	vector<int> valueVec;
	int value;
	int count=0;
```

```
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>value;
        valueVec.push_back(value);
    }
    sort(valueVec.begin(),valueVec.end());
    for(int i=0;i<valueVec.size()-1;i++){
        if((valueVec[i+1]-valueVec[i])==1){
            count++;
        }
    }
    cout<<count;
    return 0;
}
```

画图

问题描述

试题编号：	201409-2
试题名称：	画图
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 在一个定义了直角坐标系的纸上，画一个(x1,y1)到(x2,y2)的矩形指将横坐标范围从 x1 到 x2，纵坐标范围从 y1 到 y2 之间的区域涂上颜色。 下图给出了一个画了两个矩形的例子。第一个矩形是(1,1) 到(4, 4)，用绿色和紫色表示。第二个矩形是(2, 3)到(6, 5)，用蓝色和紫色表示。图中，一共有 15 个单位的面积被涂上颜色，其中紫色部分被涂了两次，但在计算面积时只计算一次。在实际的涂色过程中，所有的矩形都涂成统一的颜色，图中显示不同颜色仅为说明方便。



给出所有要画的矩形，请问总共有多少个单位的面积被涂上颜色。

输入格式

输入的第一行包含一个整数 n，表示要画的矩形的个数。

接下来 n 行，每行 4 个非负整数，分别表示要画的矩形的左下角的横坐标与纵坐标，以及右上角的横坐标与纵坐标。

输出格式

输出一个整数，表示有多少个单位的面积被涂上颜色。

样例输入

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

样例输出

```
15
```

评测用例规模与约定

1<=n<=100，0<=横坐标、纵坐标<=100。

```
#include<iostream>
#include<map>
#include<string>
#include<sstream>
using namespace std;
void int2str(const int &int_temp,string &string_temp)
{
	stringstream stream;
	stream<<int_temp;
	string_temp=stream.str();
```

```
}
int main(){
	int n;
	map<string,int> countMap;
	typedef map<string,int>::value_type valType;
	stringstream stream;
	cin>>n;
	int ** values= new int*[n];
	for(int i=0;i<n;i++){
		values[i]=new int[4];
	}
	for(int i=0;i<n;i++){
		for(int j=0;j<4;j++){
			cin>>values[i][j];
		}
		for(int a=values[i][0]+1;a<=values[i][2];a++){
			for(int b =values[i][1]+1;b<=values[i][3];b++){
				string x;
				string y;
				int2str(a,x);
				int2str(b,y);
				string xy =x+"-"+y;
				countMap.insert(valType(xy,1));
			}
		}
	}
	cout<<countMap.size();
	for(int i=0;i<n;i++){
		delete[] values[i];
	}
	delete[] values;
	return 0;
}
```

这里需要说明的是，如果你是使用 C++ 的建议直接用 vector 实现二维数组或者是一维，方便而且安全。

```
#include<iostream>
#include<map>
#include<string>
#include<sstream>
#include<vector>
using namespace std;
void int2str(const int &int_temp,string &string_temp)
{
		stringstream stream;
```

```
        stream<<int_temp;
        string_temp=stream.str();
}
int main(){
    int n;
    map<string,int> countMap;
    typedef map<string,int>::value_type valType;
    stringstream stream;
    cin>>n;
    typedef vector<int> typeVec;
    vector<typeVec> values(n,vector<int>(4,0));
    for(int i=0;i<n;i++){
        for(int j=0;j<4;j++){
            cin>>values[i][j];
        }
        for(int a=values[i][0]+1;a<=values[i][2];a++){
            for(int b =values[i][1]+1;b<=values[i][3];b++){
                string x;
                string y;
                int2str(a,x);
                int2str(b,y);
                string xy =x+"-"+y;
                countMap.insert(valType(xy,1));
            }
        }
    }
    cout<<countMap.size();
    return 0;
}
```

字符串匹配

问题描述

试题编号：	201409-3
试题名称：	字符串匹配
时间限制：	1.0s
内存限制：	256.0MB

问题描述：	**问题描述** 给出一个字符串和多行文字，在这些文字中找到字符串出现的那些行。你的程序还需支持大小写敏感选项：当选项打开时，表示同一个字母的大写和小写看作不同的字符；当选项关闭时，表示同一个字母的大写和小写看作相同的字符。 **输入格式** 输入的第一行包含一个字符串 S，由大小写英文字母组成。 第二行包含一个数字，表示大小写敏感的选项，当数字为 0 时表示大小写不敏感，当数字为 1 时表示大小写敏感。 第三行包含一个整数 n，表示给出的文字的行数。 接下来 n 行，每行包含一个字符串，字符串由大小写英文字母组成，不含空格和其他字符。 **输出格式** 输出多行，每行包含一个字符串，按出现的顺序依次给出那些包含了字符串 S 的行。 **样例输入** Hello 1 5 HelloWorld HiHiHelloHiHi GrepIsAGreatTool HELLO HELLOisNOTHello **样例输出** HelloWorld HiHiHelloHiHi HELLOisNOTHello **样例说明** 在上面的样例中，第四个字符串虽然也是 Hello，但是大小写不正确。如果将输入的第二行改为 0，则第四个字符串应该输出。 **评测用例规模与约定** 1<=n<=100，每个字符串的长度不超过 100。

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
int main(){
```

```
    vector<string> valVec;
    string search;
    int flag=0;
    int lineNum=0;
    string line;
    cin>>search>>flag>>lineNum;
    for(int i=0;i<lineNum;i++){
        cin>>line;
        valVec.push_back(line);
    }
    for(int i=0;i<lineNum;i++){
        string temp=valVec[i];
        if(flag){
            string::size_type pos =temp.find(search);
            if(pos!=string::npos){
                cout<<temp<<endl;
            }
        }else{
            transform(search.begin(),search.end(),search.begin(),::tolower);
            transform(temp.begin(),temp.end(),temp.begin(),::tolower);
            string::size_type pos =temp.find(search);
            if(pos!=string::npos){
                cout<<valVec[i]<<endl;
            }
        }
    }
    return 0;
}
```

2014 年 3 月

相反数

问题描述

试题编号：	201403-1
试题名称：	相反数
时间限制：	1.0s

内存限制：	256.0MB
问题描述：	**问题描述** 有 N 个非零且各不相同的整数。请你编一个程序求出它们中有多少对相反数(a 和 -a 为一对相反数)。 **输入格式** 第一行包含一个正整数 N。(1 ≤ N ≤ 500)。 第二行为 N 个用单个空格隔开的非零整数,每个数的绝对值不超过1000,保证这些整数各不相同。 **输出格式** 只输出一个整数,即这 N 个数中包含多少对相反数。 **样例输入** 5 1 2 3 -1 -2 **样例输出** 2

```
#include<iostream>
#include<map>
using namespace std;
int main(){
	int n,count=0;
	map<int,int> pluMap;
	map<int,int> miuMap;
	typedef map<int,int>::value_type valType;
	cin>>n;
	for(int i=0;i<n;i++){
		int value;
		cin>>value;
		if(value>0){
			pluMap.insert(valType(value,1));
		}else{
			miuMap.insert(valType(value,1));
		}
	}
	map<int,int>::iterator iter=pluMap.begin();
	map<int,int>::iterator minIter=miuMap.begin();
	int size = pluMap.size()-miuMap.size();
	if(size>0){
		while(minIter!=miuMap.end()){
			int s = minIter->first;
			if(pluMap.count(~(s - 1))){
```

```
                count++;
            }
        ++minIter;
        }
    }else{
        while(iter!=pluMap.end()){
            int s = iter->first;
            if(miuMap.count(~s +1)){
                count++;
            }
        ++iter;
        }
    }
    cout<<count;
    return 0;
}
```

窗口

问题描述

试题编号：	201403-2
试题名称：	窗口
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 在某图形操作系统中,有 N 个窗口,每个窗口都是一个两边与坐标轴分别平行的矩形区域。窗口的边界上的点也属于该窗口。窗口之间有层次的区别,在多于一个窗口重叠的区域里,只会显示位于顶层的窗口里的内容。 当你点击屏幕上一个点的时候,你就选择了处于被点击位置的最顶层窗口,并且这个窗口就会被移到所有窗口的最顶层,而剩余的窗口的层次顺序不变。如果你点击的位置不属于任何窗口,则系统会忽略你这次点击。 现在我们希望你写一个程序模拟点击窗口的过程。 **输入格式** 输入的第一行有两个正整数,即 N 和 M。(1 ≤ N ≤ 10,1 ≤ M ≤ 10) 接下来 N 行按照从最下层到最顶层的顺序给出 N 个窗口的位置。 每

行包含四个非负整数 x_1, y_1, x_2, y_2,表示该窗口的一对顶点坐标分别为 (x_1, y_1) 和 (x_2, y_2)。保证 x_1 < x_2,y_1 2。

接下来 M 行每行包含两个非负整数 x, y,表示一次鼠标点击的坐标。

题目中涉及到的所有点和矩形的顶点的 x, y 坐标分别不超过 2559 和 1439。

输出格式

输出包括 M 行,每一行表示一次鼠标点击的结果。如果该次鼠标点击选择了一个窗口,则输出这个窗口的编号(窗口按照输入中的顺序从 1 编号到 N);如果没有,则输出"IGNORED"(不含双引号)。

样例输入

```
3 4
0 0 4 4
1 1 5 5
2 2 6 6
1 1
0 0
4 4
0 5
```

样例输出

```
2
1
1
IGNORED
```

样例说明

第一次点击的位置同时属于第 1 和第 2 个窗口,但是由于第 2 个窗口在上面,它被选择并且被置于顶层。

第二次点击的位置只属于第 1 个窗口,因此该次点击选择了此窗口并将其置于顶层。现在的三个窗口的层次关系与初始状态恰好相反了。

第三次点击的位置同时属于三个窗口的范围,但是由于现在第 1 个窗口处于顶层,它被选择。

最后点击的 (0, 5) 不属于任何窗口。

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int N,M;
    typedef vector<int> intList;
    vector<intList> area;
    typedef vector<intList>::iterator iterType;
    int clik[10][2]={0};
    cin>>N>>M;
```

```
    for(int i=0;i<N;i++){
        vector<int> line;
        line.push_back(i+1);
        int pos;
        for(int j=0;j<4;j++){
            cin>>pos;
            line.push_back(pos);
        }
        area.push_back(line);
    }
    for(int i=0;i<M;i++){
        for(int j=0;j<2;j++){
            cin>>clik[i][j];
        }
    }
    for(int i=0;i<M;i++){
        bool flag = false;
        for(int q=N-1;q>=0;q--){

    if((clik[i][0]>=area[q][1])&&(clik[i][0]<=area[q][3])&&(clik[i][1]>=area[q][2])&&(clik[i][1]<=ar
ea[q][4])){
                flag = true;
                cout<<area[q][0]<<endl;
                vector<int> temp=area[q];
                iterType p = area.begin()+q;
                area.erase(p);
                area.push_back(temp);
                break;
            }
        }
        if(!flag){
            cout<<"IGNORED"<<endl;
        }
    }
    return 0;
}
```

命令行选项

问题描述

试题编号：	201403-3

试题名称：	命令行选项
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 请你写一个命令行分析程序,用以分析给定的命令行里包含哪些选项。每个命令行由若干个字符串组成,它们之间恰好由一个空格分隔。这些字符串中的第一个为该命令行工具的名字,由小写字母组成,你的程序不用对它进行处理。在工具名字之后可能会包含若干选项,然后可能会包含一 些不是选项的参数。 选项有两类:带参数的选项和不带参数的选项。一个合法的无参数选项的形式是一个减号后面跟单个小写字母,如"-a" 或"-b"。而带参数选项则由两个由空格分隔的字符串构成,前者的格式要求与无参数选项相同,后者则是该选项的参数,是由小写字母,数字和减号组成的非空字符串。 该命令行工具的作者提供给你一个格式字符串以指定他的命令行工具需要接受哪些选项。这个字符串由若干小写字母和冒号组成,其中的每个小写字母表示一个该程序接受的选项。如果该小写字母后面紧跟了一个冒号,它就表示一个带参数的选项,否则则为不带参数的选项。例如, "ab:m:" 表示该程序接受三种选项,即"-a"(不带参数),"-b"(带参数), 以及"-m"(带参数)。 命令行工具的作者准备了若干条命令行用以测试你的程序。对于每个命令行,你的工具应当一直向后分析。当你的工具遇到某个字符串既不是合法的选项,又不是某个合法选项的参数时,分析就停止。命令行剩余的未分析部分不构成该命令的选项,因此你的程序应当忽略它们。 **输入格式** 输入的第一行是一个格式字符串,它至少包含一个字符,且长度不超过52。格式字符串只包含小写字母和冒号,保证每个小写字母至多出现一次,不会有两个相邻的冒号,也不会以冒号开头。 输入的第二行是一个正整数 N(1 ≤ N ≤ 20),表示你需要处理的命令行的个数。 接下来有 N 行,每行是一个待处理的命令行,它包括不超过 256 个字符。该命令行一定是若干个由单个空格分隔的字符串构成,每个字符串里只包含小写字母,数字和减号。 **输出格式** 输出有 N 行。其中第 i 行以"Case i:" 开始,然后应当有恰好一个空格,然后应当按照字母升序输出该命令行中用到的所有选项的名称,对于带参数的选项,在输出它的名称之后还要输出它的参数。如果一个选项在命令行中出现了多次,只输出一次。如果一个带参数的选项在命令行中出 现了多次,只输出最后一次出现时所带的参数。 **样例输入**

	albw:x 4 ls -a -l -a documents -b ls ls -w 10 -x -w 15 ls -a -b -c -d -e -l 样例输出 Case 1: -a -l Case 2: Case 3: -w 15 -x Case 4: -a -b

目前提交只有 90 分，我也不知道哪里问题，头痛，如果各位大神发现，欢迎邮箱 928368598@qq.com

```
#include<iostream>
#include<string>
#include<map>
#include<vector>
using namespace std;
vector<string> split(string str,string pattern)
{
	string::size_type pos;
	vector<string> result;
	str+=pattern;
	int size=str.size();
	for(int i=0; i<size; i++)
	{
		pos=str.find(pattern,i);
		if(pos<size)
		{
			string s=str.substr(i,pos-i);
			result.push_back(s);
			i=pos+pattern.size()-1;
		}
	}
   return result;
}
bool isalnumAndMiu(string str){
	bool result;
	for(string::size_type index =0;index!=str.size();index++){
		if((islower(str[index]))||(isdigit(str[index]))||(str[index]=='-')){
			result=true;
```

```
            }else{
                result = false;
                break;
            }
        }
        return result;
}
int main(){
        string str;
        string line;
        vector<string> valVec;
        getline(cin,str);
        int lineNum;
        cin>>lineNum;
        cin.ignore();
        for(int i=0;i<lineNum;i++){
            getline(cin,line);
            valVec.push_back(line);
        }
        for(int i=0;i<lineNum;i++){
            vector<string> lineSplit = split(valVec[i]," ");
            map<string,string> resultMap;
            typedef map<string,string>::iterator iterType;
            string::size_type pos =0;
            if(lineSplit.size()>1){
                for(int j=1;j<lineSplit.size();){
                    if((lineSplit[j].size()==2)&&(lineSplit[j][0]=='-')&&(islower(lineSplit[j][1]))){
                        pos=str.find(lineSplit[j][1]);
                        if(pos!=string::npos){
                            if(pos!=str.size()-1){
                                if(str[pos+1]==':'){
                                    if(isalnumAndMiu(lineSplit[j+1])){
                                        resultMap[lineSplit[j]]=lineSplit[j+1];
                                        j+=2;
                                    }else{
                                        break;
                                    }

                                }else{
                                    resultMap[lineSplit[j]]="";
                                    j++;
                                }
                            }else{
                                resultMap[lineSplit[j]]="";
```

```
                                        j++;
                                    }
                                }else{
                                    break;
                                }
                            }else{
                                break;
                            }
                        }
                    }
                    cout<<"Case "<<i+1<<": ";
                    iterType iter=resultMap.begin();
                    for(iter;iter!=resultMap.end();iter++){
                        cout<<iter->first<<" ";
                        if(iter->second!=""){
                            cout<<iter->second<<" ";
                        }
                    }
                    cout<<endl;
                }
                return 0;
}
```

2013 年 12 月

出现次数最多的数

问题描述

试题编号：	201312-1
试题名称：	出现次数最多的数
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 给定 n 个正整数，找出它们中出现次数最多的数。如果这样的数有多个，请输出其中最小的一个。

输入格式

输入的第一行只有一个正整数 n(1 ≤ n ≤ 1000)，表示数字的个数。

输入的第二行有 n 个整数 s_1，s_2，…，s_n (1 ≤ s_i ≤ 10000，1 ≤ i ≤ n)。相邻的数用空格分隔。

输出格式

输出这 n 个次数中出现次数最多的数。如果这样的数有多个，输出其中最小的一个。

样例输入

```
6
10 1 10 20 30 20
```

样例输出

```
10
```

```
#include<iostream>
#include<map>
using namespace std;
int main(){
	map<int,int> count;
	int n;
	int max = 0;
	int maxNum=0;
	int num;
	cin>>n;
	for(int i=0;i<n;i++){
		cin>>num;
		count[num]++;
		if(count[num]>max){
			max = count[num];
			maxNum=num;
		}else if(count[num]==max){
			if(num<maxNum){
				maxNum=num;
			}
		}
	}
	cout<<maxNum;
	return 0;
}
```

ISBN 号码

问题描述

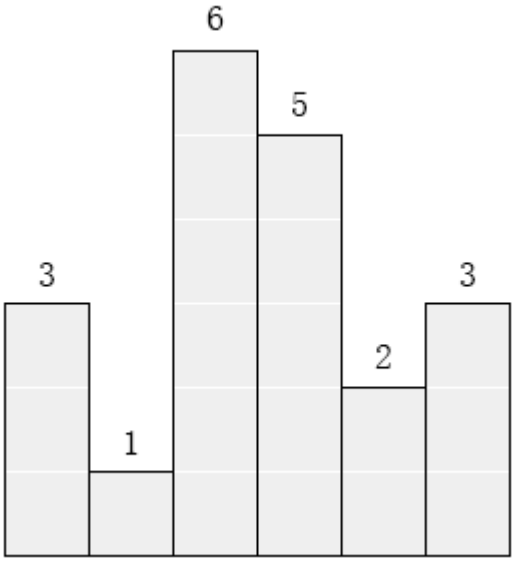
试题编号：	201312-2
试题名称：	ISBN 号码
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 每一本正式出版的图书都有一个 ISBN 号码与之对应，ISBN 码包括 9 位数字、1 位识别码和 3 位分隔符，其规定格式如“x-xxx-xxxxx-x”，其中符号“-”是分隔符（键盘上的减号），最后一位是识别码，例如 0-670-82162-4 就是一个标准的 ISBN 码。ISBN 码的首位数字表示书籍的出版语言，例如 0 代表英语；第一个分隔符“-”之后的三位数字代表出版社，例如 670 代表维京出版社；第二个分隔之后的五位数字代表该书在出版社的编号；最后一位为识别码。 识别码的计算方法如下： 首位数字乘以 1 加上次位数字乘以 2……以此类推，用所得的结果 mod 11，所得的余数即为识别码，如果余数为 10，则识别码为大写字母 X。例如 ISBN 号码 0-670-82162-4 中的识别码 4 是这样得到的：对 067082162 这 9 个数字，从左至右，分别乘以 1，2，…，9，再求和，即 0×1+6×2+……+2×9=158，然后取 158 mod 11 的结果 4 作为识别码。 编写程序判断输入的 ISBN 号码中识别码是否正确，如果正确，则仅输出“Right”；如果错误，则输出是正确的 ISBN 号码。 **输入格式** 输入只有一行，是一个字符序列，表示一本书的 ISBN 号码（保证输入符合 ISBN 号码的格式要求）。 **输出格式** 输出一行，假如输入的 ISBN 号码的识别码正确，那么输出“Right”，否则，按照规定的格式，输出正确的 ISBN 号码（包括分隔符“-”）。 **样例输入** 0-670-82162-4 **样例输出** Right **样例输入** 0-670-82162-0 **样例输出** 0-670-82162-4

```
#include<iostream>
#include<string>
using namespace std;
int main(){
	string isbn;
	int count=0;
	getline(cin,isbn);
	count=isbn[0]*1+isbn[2]*2+isbn[3]*3+isbn[4]*4+isbn[6]*5+isbn[7]*6+isbn[8]*7+isbn[9]*8+i
sbn[10]*9-48*45;
	int result = count%11;
	if(result==10){
		result =40;
	}
	if((isbn[12]-48)==result){
		cout<<"Right";
	}else{
		isbn[12]=result+48;
		cout<<isbn;
	}
	return 0;
}
```

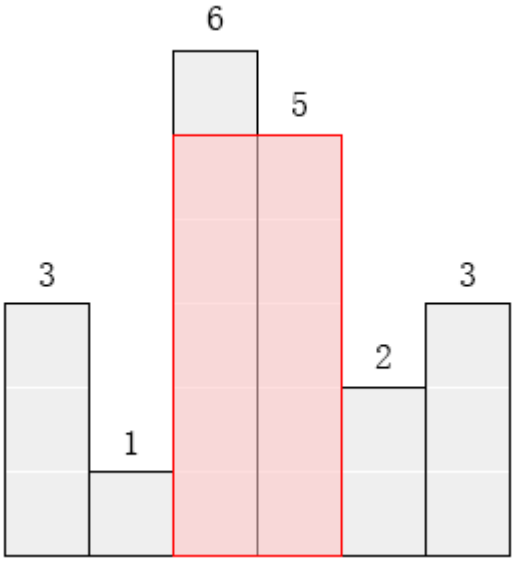
最大的矩形

问题描述

试题编号：	201312-3
试题名称：	最大的矩形
时间限制：	1.0s
内存限制：	256.0MB
问题描述：	**问题描述** 在横轴上放了 n 个相邻的矩形，每个矩形的宽度是 1，而第 i（1 ≤ i ≤ n）个矩形的高度是 h_i。这 n 个矩形构成了一个直方图。例如，下图中六个矩形的高度就分别是 3，1，6，5，2，3。



请找出能放在给定直方图里面积最大的矩形，它的边要与坐标轴平行。对于上面给出的例子，最大矩形如下图所示的阴影部分，面积是 10。



输入格式

第一行包含一个整数 n，即矩形的数量(1 ≤ n ≤ 1000)。

第二行包含 n 个整数 $h_1, h_2, \cdots, h_n$，相邻的数之间由空格分隔。(1 ≤ h_i ≤ 10000)。h_i 是第 i 个矩形的高度。

输出格式

输出一行，包含一个整数，即给定直方图内的最大矩形的面积。

样例输入

```
6
3 1 6 5 2 3
```

	样例输出 10

```
#include<iostream>
#include<string>
#include<map>
#include<vector>
#include<algorithm>
using namespace std;
typedef pair<int,int> PAIR;

bool isBig(PAIR lhs,PAIR rhs){
	return lhs.second<rhs.second;
}
int main(){
	int n=0;
	int value =0;
	vector<int> val;
	map<int,int> maxMap;
	typedef map<int,int>::value_type valType;
	typedef pair<int,int> PAIR;
	cin>>n;
	for(int i=0;i<n;i++){
		cin>>value;
		val.push_back(value);
	}
	for(int i=0;i<n;i++){
		int rN=0,lN=0,area=0;
		for(int j=i;j<n;j++){
			if(val[i]<=val[j]){
				rN++;
			}else{
				break;
			}
		}
		for(int j=i;j>=0;j--){
			if(val[i]<=val[j]){
				lN++;
			}else{
				break;
			}
		}
		area = val[i]*(rN+lN-1);
		maxMap.insert(valType(i,area));
```

```
    }
    vector<PAIR> maxVec(maxMap.begin(),maxMap.end());
    vector<PAIR>::iterator iter = max_element(maxVec.begin(),maxVec.end(),isBig);
    cout<<iter->second;
    return 0;
}
```

由于能力有限，仅仅做了前三道题目，希望能够帮助大家