2014 第六次上机解题报告

--------14211065 于济凡

目录

写在前面：

老天让你等，是为了让你等到对的人；题让你做不出来，是为了让你在自己的不断学习中提高自己，直到完成。

第一题：jhljx 学数组

Problem Description

终于到数组啦。。呵呵呵。。
jhljx 决定开始学习数组了。。
一天老师给了他一个字符串。。字符串吖。。能吃吗？
老师让你统计字符串中每个字符的个数。是不是很简单吖。

Input

输入多组数据。
每组数据为一个字符串，保证字符串的长度小于 1000。字符串中保证只有英文字母。

Output

输出字符串中出现的字符的个数。请按照先输出小写字母的个数，再输出大写字母的个数的顺序输出。每组数据间用一个空行隔开。详细请见样例。

Sample Input

AbccbA
abcdeBCCDE

Sample Output

```
 #  #  #
 #  #  #
[b][c][A]

                  #
 #  #  #  #  #  #  #  #  #
[a][b][c][d][e][B][C][D][E]
```

Hint

C 语言使用字符串处理的库函数需要添加 string.h 头文件
即#include<string.h>
C 语言请用以下形式输入

```
char a[1010];
while(gets(a))
{
    int k=strlen(a);//计算字符串长度的函数,k 表示字符串的长度
    ……
```

}

C++语言使用字符串处理的库函数需要添加 string.h 头文件或者 cstring 头文件
即#include<cstring>或者#include<string.h>都可以
C++语言请用以下形式输入
```
char a[1010];
while(cin>>a)
{
    int k=strlen(a);//计算字符串长度的函数,k 表示字符串的长度
    ……
}
```
注意每组数据的输出结果之间有一个空行。
未出现的字符不要输出。

解题思路：这道题是属于所谓的“柱状图”问题。但是这道题的难点在于不是直接以横向的方式输出柱状图，而是以纵向，于是整个解题步骤分以下几步：

1. 统计每个字符都出现过的次数 m
2. 找出其中出现次数最多的（一共横向输出 m 次）
3. 然后逐行输出“#”
4. 最后输出每一个出现过的字母。

在这样的思路下，我们就一步一步按照以上步骤开始做，但是在做这个题的时候有几个小小的技巧：

1. 可以用 ASCII 码来进行循环，这样有助于从 a 到 z,和从 A 到 Z 这样的循环
2. 输出字符数组里的字符时，可以用 printf(“[%c]”,s);这样的方法输出
3. 写代码不要烦。。。的确有点长

于是，代码如下：

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

int main()
{
    int i,m;
    char x[1001];
    while(~scanf("%s",x))
    {
        int ch[1000]={0};
        m=0;

        for(i=strlen(x)-1;i>-1;i--)
```

```
{
	ch[x[i]]++;
}

for(i='a';i<='z';i++)
{
	if(m<ch[i])
		m=ch[i];
}

for(i='A';i<='Z';i++)
{
	if(m<ch[i])
		m=ch[i];
}

while(m>1)
{
	for(i='a';i<='z';i++)
	{
		if(ch[i]==m)
		{
			printf(" # ");
			ch[i]--;
		}
		else if(ch[i]>0)
			printf("	");
	}

	for(i='A';i<='Z';i++)
	{
		if(ch[i]==m)
		{
			printf(" # ");
			ch[i]--;
		}
		else if(ch[i]>0)
			printf("	");
	}

	printf("\n");
	m--;

}
```

```
            for(i='a';i<='z';i++)
            {
                  if(ch[i]==1)
                        printf(" # ");
            }

            for(i='A';i<='Z';i++)
            {
                  if(ch[i]==1)
                        printf(" # ");
            }

            printf("\n");

            for(i='a';i<='z';i++)
            {
                  if(ch[i]==1)
                        printf("[%c]",i);
            }

            for(i='A';i<='Z';i++)
            {
                  if(ch[i]==1)
                        printf("[%c]",i);
            }

            printf("\n\n");
      }

      return 0;
}
```

第二题：巫女的怀胎

Problem Description

若非技术，那便是才能。

比如 ljx 在算术方面具有卓越的才能。
已知 x 加上 x 的各个数字之和为 y，则说明 x 和 y 是一对 cp。给出 n(1<=n<=100000)。
求 n 的最小 cp。无解输出 0。

Input

输入多组数据。（小于 100 组）
每组数据一行，为一个数 n。(1<=n<=100000)

Output

每组数据输出一行。为 n 的 cp。

Sample Input

216
121

Sample Ouput

198
0

解题思路：在最开始看这道题时候觉得这是一道难题，但是后来发现这只是一道近乎于无脑求解的问题，所以相信大家应该很快就会有思路了。

这道题只需要一些小小的简化步骤：

因为各个数位相加的和是很小的，所以其实真实的数据组数不会多大的。只要先知道输入的数字的位数 n，然后从 n*9 到 n 之间（毕竟每一个位置上最大的数字是 9），无脑地求出各个数位相加的和再加上本身，碰到相等的就输出，这就一定是最小的啦。

对了！别忘了在没有一个合适的时候要输出“0”；

于是，代码如下：

```
#include<iostream>

using namespace std;
int main()
{
	int x;
	while(cin>>x)
	{
		int i=1,counter=0,temp1=x;
		while(temp1/10!=0)
		{
			temp1=temp1/10;
			i++;
		}
		for(int t=x-i*9;t<=x;t++)
		{
			int y;
			int temp2=t,sum=0;
			while(temp2!=0)
			{
				y=temp2%10;
				temp2=temp2/10;
				sum=y+sum;
			}
			if(sum+t==x)
			{
				counter++;
				cout<<t<<endl;
				break;
			}
		}
		if(counter==0)
		{
			cout<<"0"<<endl;
		}
	}
}
```

没错，放心大胆地做吧，不会超时间。（实际上能超时的话需要的循环次数要更大）

另：不要过于害怕超时的问题，但也不能不重视，这将在第四题与第七题分别体现

第三题：善恶的彼岸

Problem Description

You guys do not notice that we are gifted just for being humans.We are absolute predators, we do not even have any enemies.Maybe there are animals watching us and thinking that "Someday we will BEAT THEM DOWN!".
但是"Someday we will BEAT THEM DOWN!"这种格式会导致让人分不出是左引号还是右引号，所以我们需要把它变为“”。
一句话：把""变为“”。

Input

输入一篇文章。

Output

输出引号转换后的文章。

Sample Input

"To be or not to be," quoth the Bard, "that is the question".

Sample Ouput

“To be or not to be,” quoth the Bard, “that is the question”.

Hint

注意是将英文引号转化为中文引号。

解题思路：这个题目，除了 Linux 系统下和 windows 系统下中文引号的不同导致对于有些同学不公平之外，其实也没有什么太大的难度。

另外：这道题我是用了一个 bool 变量来进行左右引号的判定。（毕竟在正常情况下应该先有一个左边引号再有一个右边引号吧，这一点出题者没有坑）

代码如下：

```
#include <cstdio>
#include <cstring>

int main()
{
    char p;
    bool flag=false;

    while(~scanf("%c",&p))
    {
        if(p=='"'&&flag==false)
        {
            printf("“");
            flag=true;
        }
        else if(p=='"'&&flag==true)
        {
            printf("”");
            flag=false;
        }
        else
            printf("%c",p);
    }

    return 0;
}
```

bool 变量的使用，使这道题简化了不少代码量。

这个布尔变量的应用，在本次上机题目中还会再次遇到。

第四题：jhljx 分解质因数

Problem Description

jhljx 最近在学小学数学，老师教他分解质因数。也就是说给你一个数 n,让你把它分解成若干个质数的乘积的形式。

Input

输入多组数据。
每组数据一行，为一个数 n。(1<=n<=10^6)

Output

每组数据输出一行。为一个表达式。表达式中的数字必须按从小到大的顺序排列。（详见样例）

Sample Input

11
9412

Sample Ouput

11
2*2*13*181

Hint

注意可能木有乘号哦。。
请用 scanf 和 printf 输入输出。

解题思路：这个题目其实是考察函数（判断是否为质数）（具体参见第五次上机第二题）的。

解题步骤：

1. 如果这个输入的数字本身就是质数或者 1，那么就直接输出自己
2. 如果不是的话，从 1 开始除，除到能整除的数就输出一个“*”加这个数
3. 注意第一个输出的数字不要输出“*”。

于是，代码就这么简单。（我给出的样例不够精简）

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

int sushu(int x)
{
	int counter;
		if(x==1)
		{
			counter=0;
		}
		else
		{
		int sum=0;
		for(int i=1;i<=sqrt(x);i++)
		{
			if(x%i==0)
			{
				sum=sum+1;
			}
		}
		if(sum==1)
		{
			counter=1;
		}
		else
		{
			counter=0;
		}
		}
		return counter;

}
int main()
{
	int n,counter,temp;
```

```
while(~scanf("%d",&n))
{
	counter=0;
	temp=n;
	if(sushu(n)==1||n==1)
	{
		printf("%d",n);
	}
	else
	{
		int i=2;
		while(n!=1)
		{
			if(counter==0)
			{
				if(n%i==0)
				{
					n=n/i;
					printf("%d",i);
					counter++;
					i=2;
				}
				else
				{
					i++;
				}
			}
			else
			{
				if(n%i==0)
				{
					n=n/i;
					printf("*%d",i);
					i=2;
				}
				else
				{
					i++;
				}
			}
		}
	}
	printf("\n");
}
```

}
我这个代码写的有点瘦。不美观。

第五题：jhljx 学排列组合

Problem Description

jhljx 最近又学习了一些排列组合的知识。比如什么隔板法，插空法神马的。。相信你们都会吧。。毕竟 College Entrance Examination 的数学考试中考过。。
于是 jhljx 决定来考考你。。你要是记不住的话去找高中数学老师吧。。~\(≧▽≦)/~啦啦啦你看我把题目的背景都告诉你了。。相信你能 AC 哟。。
快到圣诞节了。。于是松辰学妹送给他了一些糖果。。233
松辰学妹总共送了 n 种糖果。每种糖果的个数为 a[1],a[2],a[3]……,a[n]。jhljx 讨厌连续吃两块相同种类的糖。。所以他会尽量避免吃到同种类的糖。
为啥捏？因为他是强迫症吖。。上一次上机他的强迫症不就犯了？请猛戳这里->jhljx 的强迫症
所以请你判断他是否会吃到同种类的糖。

Input

输入多组数据。
每组数据两行，第一行为一个数 n(1<=n<=10000)，第二行为 n 个数，每个数 a[i]表示 i 种类的糖共有多少个。(保证 a[i]在 int 范围内)

Output

每组数据输出一行。如果可以避免吃到同种类的糖，输出 YeS,如果无法避免，输出 No。

Sample Input

3
4 1 1

Sample Output

No

解题思路：这道题可能是整个上机中最简单的一道题了吧。表面上看需要很多高级的方法，比如插空法等各种高中排序方法，但这道题的意思并不是要我们找出排列组合的情况数，而是要我们判断是否可以相同的不相邻。

那么就好做多啦，我们不难想到，当有某种糖的数量超过总数一半了的话（分总数奇偶讨论），就不论怎么放，都至少会有相邻的。所以我们只需要那数组储存下各种糖的数目，然后一个循环比较与总数的数量关系，就可以得出答案。

注意 1：要求输出的不是 Yes，而是 YeS。（最后一个字母大写。论防 AK 的招数 1）

注意 2：测试数据超过了 int，所以只声明了 int 范围会错（论防 AK 的招数 2）

代码如下：

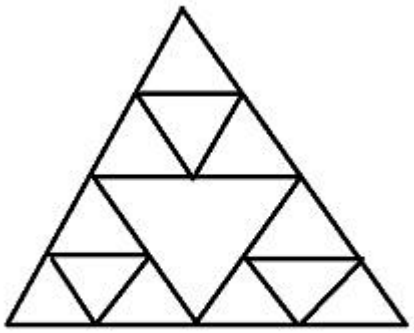
```
#include<iostream>
using namespace std;

int main()
{
	int a[100001];
	double n,sum,counter,ave;
	int i;
	while(cin>>n)
	{
		i=1;
		counter=0;
		for(;i<=n;i++)
		{
			cin>>a[i];
		}
		sum=0;
		for(int x=1;x<=n;x++)
		{
			sum=a[x]+sum;
		}
		ave=(sum+1)/2;
		for(int j=1;j<=n;j++)
		{
			if(a[j]>ave)
			{
				counter++;
			}
		}
```

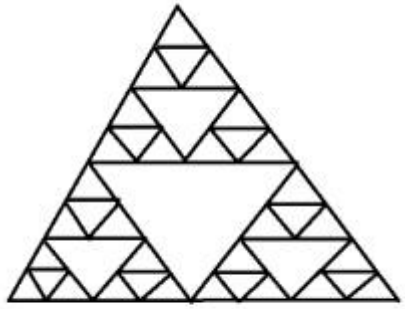
```
            if(counter==0)
            {
                cout<<"YeS"<<endl;
            }
            else
            {
                cout<<"No"<<endl;
            }

        }
}
```

第六题：jhljx 学画画

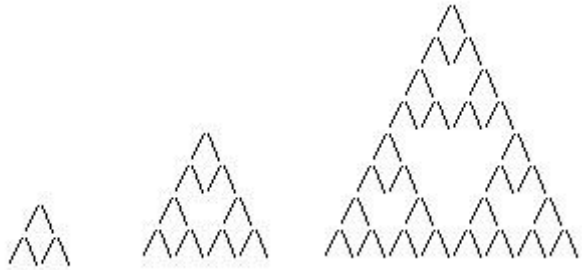
Problem Description

jhljx 是一个爱好广泛的人。最近他迷恋上了画画。还记得上次他教松辰学妹学画画。这次他决定自己动手画画啦。。
真是好棒耶。。~\(≧▽≦)/~啦啦啦
他要画的图形是这样的。。





但是为了简化，我们将图形变成了这样。



Input

输入多组数据。
每组数据输入一个 n，表示图形的行数。(保证 n 为 2,4,8,16……，且 n<=10000)

Output

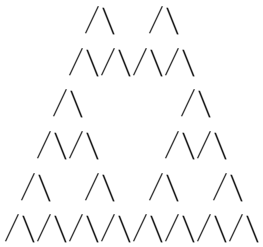
输出正确的图形。

Sample Input

8

Sample Output

```
   /\
  /\/\
```



声明：本题思路鸣谢寇宇增。

虽然前面的题也有一些难度，但是这道题与后面的那道第七题还是明显要胜于之前的所有题目的。

在开始做这道题之前，请先把自己关于三角形的知识总结一下。因为这道题可能会用到（仅限本方法）。

画图问题其实归根到底，是横向纵向的循环同时涵盖数学知识的问题，对于这种问题，我们需要找到各个位置所画图形与它的位置坐标的某些关联。

是否还记得我们之前学过的一个著名三角形：杨辉三角。

本题的思路也正是从此开始。（建议自己拿笔在纸上画一画）

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

这是到 4 次方的杨辉三角模型，让我们看看这个图形与我们的样例输出有什么相似呢？

没错！如果把所有其中的奇数的位置都画成“/\”呢？那就会变成我们想要的图形啦！

如果你愿意尝试，可以发现 8 的时候也满足此关系，16 也是！

所以呢，我们不需要知道杨辉三角的具体数值，只需要知道这个位置是奇数还是偶数就可以了。也就是将这个图形变为

```
          1
        1   1
      1   0   1
    1   1   1   1
```

这回，用上了我们离散数学中的一个异或运算就可以解决这个 0 与 1 判断的问题。（只要判断数字的位置上面两个数字是不是一样即可）

注意：

1. 需要使用二维数组。（分别为横，纵坐标）
2. 使用 bool 变量

```
#include <cstdio>
```

```
int main()
{
    int n;
    while (~scanf("%d", &n ))                    //输入数据
    {
        bool z[ n ] [ n ];                       //声明一个只能为 0 或 1 的 bool 二维数组
        bool t;
        for( int a = 1; a < n + 1; a ++)
        {
            for( int b = 1; b < a + 1; b ++)
            {
                if ( b == 1 || b == a)
                    z [ a - 1 ] [ b - 1]= 1;
                else
                {
                    t = z [ a - 2] [ b - 2] ^z [ a - 2] [ b - 1];
                    z [ a - 1] [ b - 1] = t;
               }                                 //将每一个位置都赋上 0 或 1
                }
        }
        for ( int a = 1; a < n + 1; a ++)
        {
            for ( int num = 0; num < n - a; num ++)
            {
                printf(" ");
            }
            for ( int b = 1; b < a + 1; b ++)
            {
                if ( z [ a - 1] [ b - 1] )       //布尔量为 1 时画为“/\”
                    printf("/\\");
                else
                    printf("  ");
            }
            printf("\n");                        //画出图形：有的画成“ ”有的画成“/\”
        }
    }
    return 0;
}
```

本题还有一个隐藏的陷阱：那就是没有的地方要输出“ ”。

本题小结：

二维数组的利用之一：作为横纵坐标的不断移动

另：有关数组的问题，在下一题中还将有所体现。

第七题：冰点

Problem Description

A human baby when will they find out?
That at a point they are born We are
winners of Earth.

给定从 2000 年 1 月 1 日逝去的天数，
请输出这一天的具体日期。

Input

多组数据，每组一个整数 n(n>=0) 保证结果不超过 9999 年（保证数据量小于 1e6)

Output

输出具体日期，格式为"year-month-day dayofweek”
需要前导 0

Sample Input

1
30
365
366

Sample Ouput

2000-01-02 Sunday
2000-01-31 Monday

2000-12-31 Sunday
2001-01-01 Monday

Hint

"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"
难题，慎入

注意 hint 的最后一句话。这也是我想说的。

这道题的题干，令许多同学误以为是以前曾经做过的日期判断。实际上，它的运算比之前已知日期的题目要复杂得多。

而且，这道题作为一道杭电 OJ，POJ,ZOJ 都曾出现过的题目，是一道 ACM 入门的经典习题，其做法可以在一定程度上起到磨练算法思路的作用。

其经典的解法代码如下：

```
#include<cstdio>

char week[7][10] = {"Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"};
int year[2]= {365, 366};
int month[2][12] = {31, 28, 31, 30, 31, 30, 31, 31, 30,31, 30, 31,31, 29, 31, 30, 31, 30, 31, 31, 30,31, 30, 31};                    //定义了年，月，日的数组。其中星期几的字符数组单纯为了简洁，可以在后期用 switch 代替。
int f(int m)
{
    if(m%4!=0||(m%100==0&&m%400!=0))
    {
        return 0;
    }
    else
    {
        return 1;
    }
}                                                                    //本函数为是否为闰年的判断。
int main()
{
    int n,i,j,day,spyear;
    while(~scanf("%d",&n))
    {
        day=n%7;
        for(i=2000;n>=year[f(i)];i++)        //使年数 i 不断增加，每次根据是否闰年将总日数减去 365 或 366
        {
            n=n-year[f(i)];
        }
```

```
            for(j=0;n>=month[f(i)][j];j++)                //同样的，每次减去该月的日数目
            {
                n=n-month[f(i)][j];
            }
            printf("%d-%02d-%02d %s\n",i,j+1,n+1,week[day]);      //输出结果
        }
}
```

小技巧：输出 01 等可以采用”%02d”这样的输出格式，保证了输出的合乎规范

但是!! 这样的路数在北航不通!!

因为北航的测试是有时间限制的，1000ms 才是这道题目的真正难点。

如何才能提高速度呢？
想到了北航第五次上机的最后一题，有一个用除法代替减法从而提高效率的方法。

那么，我们的年分计算，在这里是每次加 1，可不可以用除法乘法一次性搞定呢？
当然可以。
我们可以这样做。先看看这些日子包括多少个 400 年，再判断包括多少个 100 年，然后判断有多少个 4 年，最后，逐年递减。

这样的话，我们就只需计算聊聊几次，循环次数很少了。
在判断有多少个 400 年，100 年，4 年时，要注意其中包含的闰年数。
同时，有个扰乱我们数据的东西，那就是 2000 年本身是闰年，那多不好判断啊，我们不妨计算年分的时候从 2001 年开始计算，这样就回避了讨论。
（毕竟 2000 年不需要计算年分了）

代码如下：

```
#include<cstdio>

char week[7][10] = {"Saturday", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"};
int year[2]= {365, 366};
int century[2]={36524,36525};          //以闰年开头的世纪会多一天
int month[2][12] = {31, 28, 31, 30, 31, 30, 31, 31, 30,31, 30, 31,31, 29, 31, 30, 31, 30, 31, 31, 30,31, 30, 31};
int f(int m)
{
    if(m%4!=0||(m%100==0&&m%400!=0))
    {
        return 0;
    }
    else
```

```
    {
        return 1;
    }
}
int main()
{
    int n,i,j,day;
    while(~scanf("%d",&n))
    {
        day=n%7;
        int i = 2000;
        if(n>365)                    //计算年份时从 2001 年开始计算，不考虑天数在 365 以内的情况
        {
            int temp;
            n -= 366;
            i++;
            temp = n / ( 365 * 400 + 97 );
            i += temp* 400;
            n -= ( 365 * 400 + 97 ) * temp;
            for(;n>=century[f(i+99)];i+=100)
            {
                n-=century[f(i+99)];          //计算不同世纪时候不能单纯地去减，毕竟不同的世纪有所不同
            }
            temp= n / ( 365 * 4 + 1 );
            i += temp * 4;
            n -= ( 365 * 4 + 1 ) * temp;
            for(;n>=year[f(i)];i++)          //最后逐年递减时还像刚才一样
            {
                n-=year[f(i)];
            }
        }
        for(j=0;n>=month[f(i)][j];j++)
        {
            n=n-month[f(i)][j];
        }

        printf("%d-%02d-%02d %s\n",i,j+1,n+1,week[day]);
    }
}
```

几点说明：

1. 在执行过程中，temp（temporary）变量完全是先代表几个 400 年，几个 100 年等等。

2．所谓的减少时间，就是某种程度上用人脑代替计算机进行运算。

3．本题思考过程当中，其实有很多地方可以用打补丁的方法做出来，但是这样就不能体会到这道题的好处来了，所以我还是没有选择打补丁。

本题很不错。如果大家有好的方法，希望能够与我分享。