第五次练习解题告

------by 14211062 杨晨

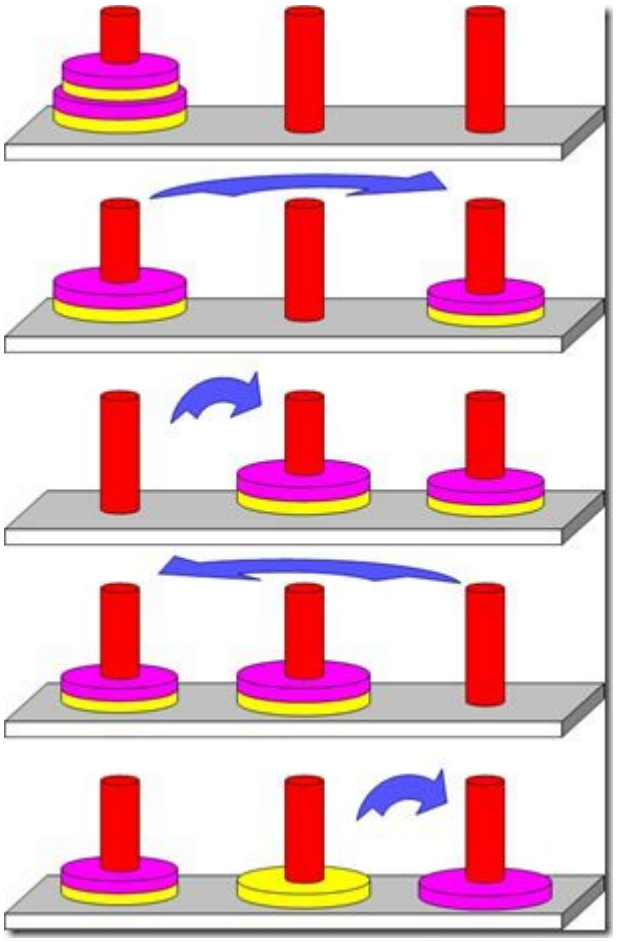
题目 A：双层汉诺塔

Problem Description

想必，前面的汉诺塔对于大家来说，just a piece of cake!

我们来玩一个更高级的汉诺塔游戏——双层汉诺塔。双层汉诺塔游戏中，相同大小的盘都有颜色不同的两片，按照次序放在最左边的金刚柱上。
游戏的目的是把不同颜色的盘分别按照上小下大的顺序放在中间和右边的两个柱子上。在移动过程中，在移动过程中，
依旧是遵守大盘必须在小盘之下，而颜色顺序无限制。

下面给出的四个盘子的例子，当然，这个图片是不完整的，还需要把左边粉红色的盘子移到最右边的柱子上，把黄色的盘子移动到中间的柱子上。



Input

第一行输入一个整数 t（t < 10），表示测试的组数。
接下来的 t 行，每行输入一个正偶数 n（0 < n <=20）,表示盘子的个数。

Output

对于每组测试数据，首先输出一行 Case #X：，表示第 X 组测试数据。
接下来，每行都输出移动盘子的步骤，对于每次移动，以"Move from X to Y"表示，表示需要将 X 柱子上最顶层的盘子移动到 Y 柱子上，其中 X，Y 是 A,B,C 中的一个，当然，X 和 Y 不可能相同

Sample Input

```
2
2
4
```

Sample Output

```
Case #1:
Move from A to B
Move from A to C
Case #2:
Move from A to C
Move from A to C
Move from A to B
Move from A to B
Move from C to A
Move from C to A
Move from B to C
Move from A to B
Move from A to C
```

解析：

这道题与单层汉诺塔一样，也要用到递归。若想完成 n 块，需要先将 n-2 块借助 B 移到 C，再将最后 2 块移到 B，之后将 n-2 块借助 B 移回 A，再将 B 上的 2 块中的一块移到 C，此时最后 2 块就完成了。而若想完成剩下的 n-2 块，则需再由 n-4 块和 2 块重复上面的操作。之后层层下放，直至最后 2 块。完成 n-2 块的移动需要构造两个函数 hanoi1 和 hanoi2，前者负责将 n-2 块从 A 借助 B 移动

到 C，后者负责将 n-2 块从 C 借助 B 移回 A，在这二者之间还有一个将最后两块从 A 移到 B 的过程。此外，还需要构造一个主函数 hanoi,它负责调用 hanoi1 和 hanoi2 完成前 n-2 块的移动，以及递归地调用自己来完成所有步骤。

代码：

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    void hanoi(int,char,char,char);//主函数
    int t,n;
    scanf("%d",&t);
    for(int i=1;i<=t;i++)
    {
        scanf("%d",&n);
        printf("Case #%d:\n",i);
        hanoi(n,'A','B','C');
    }
}
void hanoi(int n,char one,char two,char three)
{
    void hanoi1(int,char,char,char);//前 n-2 块从 A 到 C
    void hanoi2(int,char,char,char);//前 n-2 块从 C 到 A
    if(n==2)
        {
            printf("Move from A to B\n");
            printf("Move from A to C\n");
        }
    else
        {
            hanoi1(n-2,one,two,three);//调用 hanoi1 实现前 n-2 块从 A 到
C
            printf("Move from A to B\n");
            printf("Move from A to B\n");//最后两块从 A 到 B
            hanoi2(n-2,three,two,one);//调用 hanoi2 实现前 n-2 块从 C 到
A
            printf("Move from B to C\n");
            hanoi(n-2,one,two,three);//递归地自己调用自己
        }
}
void hanoi1(int n,char one,char two,char three)//实现前 n-2 块从 A 到 C
    {
        void move1(char x,char y);
```

```
        if (n==2)
            {
                move1(one,three);
            }
        else
        {
            hanoi1(n-2,one,three,two);//递归地自己调用自己
            move1(one,three);
            hanoi1(n-2,two,one,three);//递归地自己调用自己
        }
    }
    void hanoi2(int n,char three,char two,char one)//实现前 n-2 块从 C
到 A
    {
        void move1(char x,char y);
        if (n==2)
            {
                move1(three,one);
            }
        else
        {
            hanoi2(n-2,three,one,two);//递归地自己调用自己
            move1(three,one);
            hanoi2(n-2,two,three,one);//递归地自己调用自己
        }
    }
    void move1(char x,char y)//移动函数，模拟移动过程
    {
        printf("Move from %c to %c\n",x,y);
        printf("Move from %c to %c\n",x,y);
    }
```

题目 B：Two Bags of Chocolates

Problem Description

Darkness Wong 有两袋巧克力，已知第一袋里有 x(x>=1)块，第二袋里有(y>=1)块。

很多周以前我们就知道，Darkness 是一个很粗心的人，毫不意外地，他把第一袋巧克力丢了。

Darkness 还记得两袋巧克力总块数(x+y)的一些特征。首先，(x+y)不大于一个特定的正整数 N；其次，(x+y)可以被正整数 k 整除。

帮助 Darkness 计算出第一袋中可能的巧克力块数 x。请升序输出所有可能的结果。

Input

多组测试数据。

每行测试数据只有一行，三个正整数 y，k，n(1<=y,k,n<=10^9;(n/k)<=10^5)。

Output

对于每组数据，输出一行，升序输出 x 所有可能的结果。

如果不存在满足条件的 x，输出-1。

Sample Input

```
10 1 10
10 6 40
```

Sample Output

```
-1
2 8 14 20 26
```

解析：

这道题思路很简单。只要把情况讨论全就可以过了。
（1）当 n-y<=0 的时候，所得的 x 不满足 x>=1，所以此种情况为无结果，输出“-1”；
（2）当 n-y==1 的时候，供选择的 x 值只有 x=1,此时 x+y=n.则要考虑第二个条件 x+y 能被 k 整除。也就是说如果 n%k==0,那么 x=1 为一个符合条件的结果；否则无结果，输出“-1”；
（3）当 n-y>1 的时候，从 2 到 n-y 的所有数都在考虑范围之内，此时用循环验证之间的每个数，若满足条件（x+y）%k=0，则输出；若所有数都不满足条件，输出“-1”.

代码：

```
#include<iostream>
using namespace std;
int main()
{
    long long  y,k,n;
    while(cin>>y>>k>>n)
    {
        if((n-y)<=0)
        cout<<"-1"<<endl;//(n-y)<=0 的情况
        else if((n-y)==1)//(n-y)==1 的情况
        {
            if((n%k)==0)
                cout<<"1"<<endl;//满足条件
            else
                cout<<"-1"<<endl;//不满足条件
        }
        else
        {
            int counter=0;//计数器，计算符合条件的数的个数
            for(long long i=1;i<=(n-y);i++)
            {
                if(((i+y)%k)==0)
                {cout << i <<" ";
                counter++;}
            }//循环验证，符合条件输出；
            if(counter==0)
                cout<<"-1";//计数器为 0，表示无符合条件的数；
            cout<< endl;
        }
    }
}
```

题目 C：圆有点挤

描述

gg 最近想给女友送两个精美的小礼品：两个底面半径分别为 R_1 和 R_2 的圆柱形宝石，并想装在一个盒子里送给女友。好不容易找到了一个长方体的盒子，其底面为 A*B 的矩形，他感觉好像宝石装不进去，但又不敢轻易塞进去试试。现请你帮他判断两个宝石能否放进盒子里(宝石只能竖直放置，且不能堆叠)。

输入

输入的第一行是一个整数，为数据的组数 t（t<=1000）。

每组数据占一行，包括 4 个数 A，B，R_1，R_2，均为不超过 10^4 的正整数。

输出

对于每组数据，若两个宝石能放进盒子中，则输出 YES，否则输出 NO。

样例输入

```
2
10 10 1 1
10 10 4 4
```

样例输出

```
YES
NO
```

解析：

这道题分两种情况讨论，如果矩形中有一个圆柱放不进去，即 2R1 或 2R2 大于 A 或 B 结果必然是失败；如果两个圆柱都能放进去，只需满足 (A-(R1+R2))^2+(B-(R1+R2))^2>=(R1+R2)^2 即可。

代码：

```
#include<iostream>
using namespace std;
int main()
{
    int n,A,B,R1,R2,x,y;
```

```
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>A>>B>>R1>>R2;
        if((2*R1)>A||(2*R2)>A||(2*R1)>B||(2*R2)>B)
            cout<<"NO"<<endl;//有一个圆柱放不进去
        else//两个都能放进去
        {
        x=(A-(R1+R2))*(A-(R1+R2))+(B-(R1+R2))*(B-(R1+R2));
        y=(R1+R2)*(R1+R2);
        if(x>=y)
            cout<<"YES"<<endl;//满足条件
        else
            cout<<"NO"<<endl;//不满足条件
        }
    }
}
```

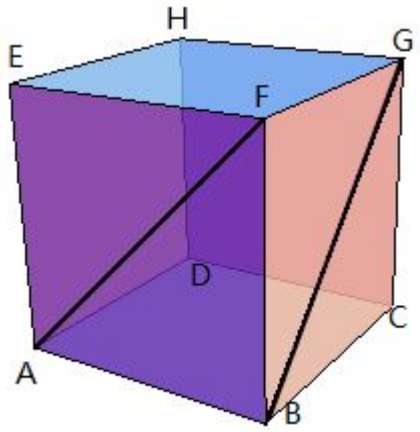
题目 D：晴天小猪的妹妹

Limit

Time Limit : 2 s
Memory Limit : 65535 KB

Description

晴天小猪是一个正儿八经的人。
但是晴天小猪也有妹子，还是个萝莉。
这天，小萝莉要高考了，他跑过来问哥哥一个题目：



长方体 ABCD-EFGH 中，AD=x，DC=y，AE=z， 问直线 AF 和直线 BG 之间的距离是多少?

晴天小猪不会，他来求助于你。

Input

第一行一个整数 T ， 为数据组数。
以下 T 组测试数据。
每组测试数据一行，为 3 个整数 x,y,z (1 <= x,y,z <= 10000)

Output

每组测试数据输出一行，表示两直线之间的距离。保留小数点后 5 位有效数字。

Sample Input

```
1
1 1 1
```

Sample Output

```
0.57735
```

解析：

这道题运用空间向量可以得出两条直线间距离公式：
D=x*y*z/sqrt((x*y)^2+(x*z)^2+(y*z)^2);
有了公式解题应该不成问题。
此外应注意：

（1）x,y,z,d 等变量要声明为 double 类型，否则总是 Wrong Answer.
（2）题目中要求保留 5 位小数，需要格式控制，不要忘了头文件<iomanip>;
（3）题中用到<cmath>库函数 sqrt,不要忘了头文件。

代码：

```
#include<iostream>
#include<iomanip>
#include<cmath>
using namespace std;
int main()
{
    int T;
    double x,y,z;double distance;//注意要声明为 double
    cin>>T;
    for(int i=1;i<=T;i++)
    {
        cin>>x>>y>>z;
        distance=(x*y*z)/sqrt((x*y)*(x*y)+(x*z)*(x*z)+(y*z)*(y*z));
        cout<<setprecision(5)<<fixed<<distance<<endl;//格式控制，保留
5 位小数
    }
}
```

题目 E：伤心的晴天小猪

Problem Description

上次上机后，好多同学居然责怪可爱的晴天小猪。苯苯的晴天小猪没有办法，遇到各种问题只能找聪明的你们求助。既然大家都不喜欢晴天小猪，伤心的他决定乘坐时光机去另一个时间点，摆脱这个伤心地。但是这个时光机太奇怪了，居然可以随意设定时间，10 月 50 日都能设定，到达这个时间点估计就灰飞烟灭了。可是苯苯的晴天小猪不知道哪些时间是存在的，于是他只能再次向聪明的你们求助了。相信善良的你们不会故意让他灰飞烟灭的吧^_^（画外音：就是要让他永远消失→_→……）

Input

第一行是数据组数 n，接下来 n 组数据，每组数据一行按照年，月，日，小时，分钟，秒给定一个时间。（20：00：00 将以 20:0:0 给出，详见样例）

Output

如果这个时间点合法（不考虑闰秒-_-||）请输出 Good bye!否则请输出 Please wait for a moment!

Sample Input

```
2
2011 11 2 19 30 0
2011 2 29 1 20 35
```

Sample Output

Good bye!
Please wait for a moment!

解析：

这道题思路很简单，就是逐级讨论，只要时间不合法就输出“Please wait for a moment!”,只有所有时间都合法才输出“Good bye!”.
都有哪些时间不合法呢？
（1）年份只要>=0 就合法；
（2）月份只要>=1 且<=12 就合法；
（3）日期如果<=0 肯定不合法；
　　如果>=1 的话：
　　每年的 1、3、5、7、8、10、12 月，日期>31 不合法；
　　每年的 4、6、9、11 月，日期>30 不合法；
　　闰年的 2 月，日期>29 不合法；
　　平年的 2 月，日期>28 不合法。
　闰年：能被 4 整除但不能被 100 整除的年份/能被 400 整除的年份；
　平年：不能被 4 整除的年份/能被 100 整除但不能被 400 整除的年份。
（4）小时>=0 且<=23 合法，否则不合法；
　　分钟>=0 且<=59 合法，否则不合法；
　　秒数>=0 且<=59 合法，否则不合法。
用以上验证条件写 if 嵌套结构就可以了。

代码：

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    long long n,y;
    int m,d,h,mi,s;
    scanf("%lld",&n);
    for(long long i=1;i<=n;i++)
    {
    scanf("%d %d %d %d %d %d",&y,&m,&d,&h,&mi,&s);
    if(y>=0)//年份合法
    {
    if(m>12 || m<1)//月份不合法
    printf("Please wait for a moment!\n");
    else//月份合法
    {
```

```
    if(m==2)//2 月的情况
    {
    if(((y%4)!=0)||((y%100)==0 && (y%400)!=0))//平年 2 月
    {
    if(d>=29||d<1)//日期不合法
    printf("Please wait for a moment!\n");
    else//日期合法
    {
    if (h>=0&&h<=23&&mi>=0&&mi<=59&&s>=0&&s<=59)//时分秒合法
    printf("Good bye!\n");
    else//时分秒不合法
    printf("Please wait for a moment!\n");
    }
    }
    else if(((y%4)==0 && (y%100)!=0)||((y%400)==0))//闰年 2 月
    {
    if(d>=30||d<1)//日期不合法
    printf("Please wait for a moment!\n");
    else//日期合法
    {
    if (h>=0&&h<=23&&mi>=0&&mi<=59&&s>=0&&s<=59)//时分秒合法
    printf("Good bye!\n");
    else//时分秒不合法
    printf("Please wait for a moment!\n");
    }
    }
    }
    else if(m==1 || m==3 || m==5 || m==7 || m==8 || m==10 || m==12)//31
天月的情况
    {
    if(d>=32||d<1)//日期不合法
    printf("Please wait for a moment!\n");
    else//日期合法
    {
    if (h>=0&&h<=23&&mi>=0&&mi<=59&&s>=0&&s<=59)//时分秒合法
    printf("Good bye!\n");
    else//时分秒不合法
    printf("Please wait for a moment!\n");
    }
    }
    else//30 天月的情况
    {
    if(d>=31||d<1)//日期不合法
    printf("Please wait for a moment!\n");
```

```
    else//日期合法
    {
    if (h>=0&&h<=23&&mi>=0&&mi<=59&&s>=0&&s<=59)//时分秒合法
    printf("Good bye!\n");
    else//时分秒不合法
    printf("Please wait for a moment!\n");
    }
    }
    }
    }
    else//年份不合法
    printf("Please wait for a moment!\n");
    }
}
```

题目 F：晴天小猪的绕口令

Problem Description

第二个游戏是绕口令。规则是：主持人给出一串字符串，要求把这串字母简化。该串字符串全部为小写英文字母。
比如：aaabbbcc，则简化为 3a3b2c；zzzzeeeeea，则简化为 4z5e1a。依次类推。
最后一题了，帮帮晴天小猪哦。
注意：本题禁止使用数组和字符串。否则此题记 0 分。

Input

第一行为一个整数 n，表示共有 n 组测试数据（1<=n<=100）。每组测试数据有一行，该行第一个数为字符串长度 t（ t <= 1,000,000）,然后为一行长度为 t 的字符串。

Output

对于每组输入数据输出一行，即简化后的字符串。

Sample Input

```
3
7 aaaaaaa
4 abcd
6 qwweee
```

Sample Output

```
7a
1a1b1c1d
1q2w3e
```

解析：

这道题思路上不难想。
如果字符串中只有一个字符 c，那么就输出“1c”；
如果字符串中字符个数>=2 个，需要借助循环来实现。首先现在循环外输出一个字符 c，并用字符 p 保留 c，之后在循环中输入 n-1 次字符 c.本题我设了 3 个计数器：第一个为 counter，初始化为 1，用来计算每个字符的个数，当出现不同字符时他要输出 1 次，并且重新变回 1；第二个为 time，用来计数是否出现过不同字符，初始化为 0，如果循环过程中出现了不同字符,它会自增，所以在循环结束只需根据 time 的值判断是否有不同字符出现；第三个为 icounter，用来输出最后一类字符，初始化为 1，如果循环中出现了不同字符，icounter 自动变为 0，在循环结束后可以作为输出最后一类字符的判断条件。如果 icounter 始终为 1，那么说明输入的所有字符完全一样，就可以发挥 time 的功能了。
本题内存卡得比较严，故应该用 C 语言<cstdio>,printf,scanf,getchar()等来实现，用<iostream>会爆内存。

代码：

```
#include<cstdio>
using namespace std;
int main()
{
    int n,t;char c,p;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&t);
        if(t==1)//只输入一个字符
        {
            getchar();//读入回车
            c=getchar();//读入字符
```

```
            printf("1%c",c);
        }
        else//输入 2 个以上字符
        {
        int counter=1;int icounter=1;int time=0;
        getchar();//读入回车
        c=getchar();//读入字符
        p=c;//将 c 保存在 p 中
        for(int num=2;num<=t;num++)
        {
            c=getchar();//读入字符
            if(c==p)
            counter++;//计算同类字符个数
            else
            {
                printf("%d%c",counter,p);
                p=c;
                counter=1;//重新初始化
                icounter=0;//更改 icounter 值表示已出现不同类字符
                time++;//验证是否出现不同类字符
            }
        }
        if(time==0)
            printf("%d%c",t,c);//用于所有字符相同时
        if(icounter==0)
            printf("%d%c",counter,c);//用于输出最后一类字符
        }
        printf("\n");
    }
}
```