STU2019ACM新生赛题解

比赛地址: https://vjudge.net/contest/335327 密码:stuacm2019

A - 计算A+B

题意:

计算输出a+b.

样例输入

2

1 5

10 20

样例输出

6

30

思路

签到题,适应网站交题,读入一个n为数据的组数,每组读入两个数字 以及输出回车.下面代码均为C++,学习拓展下.

代码

```
#include<iostream> //C++头文件
using namespace std; //使用标准库命名空间
int main()
{
   int N;
   cin>>N;
   while(N--)  //自己体会下
   {
       int a,b;
       cin>>a>>b; //输入流
       cout<<a+b<<endl; //输出流 endl是换行
   }
   return 0; //OJ要求程序正确退出返回0
}
```

B - 计算球体积

题意:

输入半径,输出球的体积. π=3.1415927

样例输入

1

1.5

样例输出

4.189

14.137

思路

根据$V=4\pi r^3/3$公式计算得出结果,注意下读入 判断输入到达文件结尾,注意题目要求的π的值还有要求结果保留 3 位小数.

代码

```
#include<iostream>
using namespace std;
int main()
{
   double r;
   while(cin>>r) //根据cin的返回值
       printf("%.3lf\n",4*3.1415927*r*r*r/3);
   return 0;
}
```

C - 数列求和

题意:

数列的第一项为n,以后各项为前一项的平方根, 求数列的前m项的和.

样例输入

81 4

2 2

样例输出

94.73

3.41

思路

迭代求和,根据数列前一个值可得出第二个值,求和即可. 此处需要用到库函数sqrt()来计算平方根,注意输出保留三位小数.

代码

```
#include<iostream>
#include<cmath> //数学公式的头文件
using namespace std;
int main()
{
   double n,m;
   while(cin>>n>>m)
   {
       double sum=n;
       while(--m)sum+=(n=sqrt(n)); //自己体会一下
       printf("%.2lf\n",sum);
   }
   return 0;
}
```

D - 分西瓜

题意:

输入一个整数w,判断是否能被分成两个偶数.

样例输入

8

样例输出

YES

思路

判断一下n的奇偶即可,特判2,因为2只能分成两个1. 注意输出答案是全大写字母. 至于证明一个非2偶数一定能分成两个偶数和奇数只能分成 一个偶数和一个奇数想必就不用了

代码

```
#include<iostream>
using namespace std;
int main()
{
   int n;
   cin>>n;
   if(n==2||n%2)cout<<"NO"; //n%2返回值为非0即奇数为false
   else cout<<"YES";
```

```
    return 0;
}
```

E - A的B次方

题意:

输入A和B,输出A^B的后三位,A=B=0时候不处理 且退出程序.

样例输入

2 3

12 6

6789 10000

0 0

样例输出

8

984

1

思路

- 方法1:直接B个A相乘然后取后三位.直接得出最后结果的话显然数据太大无法存下, 那么就要每步取余,有公式:`(a*b)%d==((a%d)*(b%d))%d`. 应用公式即可迭代出答案
- 方法2:快速幂算法.什么是快速幂,下面由~~*抗电卧底*~~成哥解释下

> 我们都知道每个数在计算机的储存方式其实都是二进制,010101这样来储存的,比如说 7可以写成0111. 所以你们要清楚一点,每个数字都可以写成二进制的形式. 同时,某个数的二进制也可以被写成:
>
> $$7->0111->1*2^0+1*2^1+1*2^2+1*2^3$$
>
> 也就是说,任何一个数,都可以分解成上面最右边的那种 形式(相信大家高中都学过-.-!) 所以,我们题目中所提到的A的B次方,B也可以写成上面那种形式
>
> $$A^B=A^{1*2^0+1*2^1+1*2^2+1*2^3+...+1*2^n}$$
>
> 上面的1也可以是0,取决于二进制位上有没有这个数字. 所以
>
> A^B可以拆成很多项相乘,也就是指数函数的指数相加
>
> 那么我们在算A^B的时候,只需要判断2^n的系数是否为1,如果是1 说明我要进行这个计算,如果不是1,那么我们就不用进行这个计算. 那么接下来直接看模板:

```
long long MOD;
long long quick_pow(long long a,long long b)
{
  long long ans=1;
  while(b)     //当指数不为0的时候
  {
      if(b&1)   //判断b的二进制的最后一位是否为1,也就是判断我要不要乘这一项
          ans=(ans*a)%MOD;   //累乘上此时a的x次方,x为二进制展开式
      b >>= 1;   //舍去b的二进制的最后一位
```

```
        a = (a*a) % MOD // 计算出a的x次方,x为二进制展开式
    }
  }
```

成哥说完我来补充一点好了,这里我们来对这模板代码举例3的5次方:

> 首先5的二进制是101,ans初始化时候为1
> 然后5不为0进入了while循环
> 此时101最后一位为1进入了if后ans就乘上了3 (3^{1*2^0}) 就变成了ans==3
> 接下来b右移一位就变成了10然后a就变成了9 (3^{1*2^1}) 然后再次进入while循环,这次的if可就进不去了
> b右移一位变成1,a则变成81 (3^{1*2^2}) 最后ans乘上81变成了ans==243完成计算 所以本质就是:
>
> $$A^B=A^{1*2^0+1*2^1+1*2^2+1*2^3+\ldots+1*2^n}=A^{1*2^0}*A^{1*2^1}*A^{1*2^2}*A^{1*2^3}*\ldots*A^{1*2^n}$$

代码1

```
#include<iostream>
using namespace std;
const int MOD=1000; //定义取余
int main()
{
   int A,B;
   while(cin>>A>>B&&(A||B))  //自己体会一下
   {
       A%=MOD;
       int ans=1;
       while(B--)ans=(ans*A)%MOD;
       cout<<ans<<endl;
   }
   return 0;
}
```

代码2

```
#include<iostream>
using namespace std;
const int MOD=1000; //定义取余
int quick_power(int a,int b) //快速幂的模板
{
   long long ans = 1;
       while (b) {
           if (b & 1)ans=(ans*a)%MOD;
           a = (a*a)%MOD;
           b >>= 1;
       }
       return (int)ans;
}
int main()
{
   int A,B;
   while(cin>>A>>B&&(A||B))  //自己体会一下
       cout<<quick_power(A,B)<<endl;
```

```
    return 0;
}
```

F - 母牛生小牛

题意:

有一头母牛,每年会生一头小母牛,每头小母牛在第四年长大可以生牛. 问在第n年的时候共有多少头母牛.n=0的时候结束.

样例输入

2

4

5

0

样例输出

2

4

6

思路

成哥说:

> 其实和Fibonacci数列一样,找递推式子就好了,那么这一题的递推关系怎么找呢?
> 大家可以这样子理解:
> 假设我要求第n年母牛的数量,那么这一年的母牛的来源会是哪些部分呢?
> 很显然,一部分是之前母牛,一部分是今年出生的
> 那堆之前的母牛的数量是多少呢?没错,就是n-1年去年的牛的数量 那今年出生的牛怎么算呢?首先我们知道了,每头小母牛从第四个年头开始,每年 年初也会生一头小母牛(别问为什么没有公牛,问就是不知道),也就是说,今年 生出来的小母牛,全部都是n-3年的牛生的

代码

```
#include<iostream>
using namespace std;
int main()
{
    int n, cows[56]={0,1,2,3}; //初始化前几年牛的数量,注意0下标无意义
    for(int i=4;i<=55;i++)cows[i]=cows[i-1]+cows[i-3]; //迭代计算到55年时候
    while(cin>>n&&n)cout<<cows[n]<<endl;
    return 0;
}
```

G - 杨辉三角

题意:

按格式输出杨辉三角前n行,输出每一层的整数之间用一个空格隔开,每一个杨辉三角后面加一个空行.

样例输入

2 3

样例输出

1

1 1

//空行

1

1 1

1 2 1

思路

高中杨辉三角原题,可能会在输出格式那被卡一下,杨辉三角可是个算组合数的好东西, 由高中知识或者画图找规律可以知道,第n层有n个数字,从左到右C(n,m),m=0,1,2,...,n. 那么杨辉三角最重要就是要算出每一个位置的组合数,有个重要的公式就是

`C(n,m)=C(n-1,m-1)+C(n-1,m)` 以及 `C(n,0)=C(n,n)=1`

由以上两个式子即可求出全部位置的组合数

代码

```
#include<iostream>
using namespace std;
int main()
{
   int C[31][31];
   for(int n=0;n<=30;n++)    //根据题目规模预处理
       for(int m=0;m<=n;m++)
       if(m==0||m==n)C[n][m]=1; //当m为0或n
       else C[n][m]=C[n-1][m-1]+C[n-1][m];  //迭代公式
   int n;
   while(cin>>n,n--) //因为我们的C数组从C(0,0)开始为第一行
   {
       for(int i=0;i<=n;i++){
           for(int j=0;j<=i;j++)
               if(j!=0)cout<<" "<<C[i][j];  //注意下输出格式
               else cout<<C[i][0];
           cout<<endl;
       }
       cout<<endl;  //输出格式有空行
   }
   return 0;
}
```

H - LCM PLUS

题意:

有多组测试样例,求多个数的最小公倍数

样例输入

2 4 6

3 2 5 7

样例输出

12

70

思路

我们知道,两个数的最小公倍数的求法公式是:

$$LCM(a,b)=a*b/GCD(a,b)$$

例如我们已经知道6和10的最大公约数是2,那么6和10的最大公倍数就是 6*10/2=30. 然后可以推广到多个数的最小公倍数求法为:

$$LCM(a,b,c)=LCM(LCM(a,b),c)$$

$$LCM(a,b,c,d)=LCM(LCM(LCM(a,b),c),d)$$

例如6和10和7的最小公倍数是30和7的最小公倍数为30*7/1=210;6和10和7和21的最小公倍数是210*21/21=210. 知道LCM的求法之后就只剩下GCD的求法了. GCD我们最常用的算法是辗转相除法,下面给出模板:

```
int GCD(int a,int b)
{
   if(a<b)return GCD(b,a); //保证a比b大,如果在一开始传进来的参数已经保证则可以不需要
   if(!b)return a; //当b为0的时候证明已经能够整除
   return GCD(b,a%b); //辗转相除法
}
```

代码

```
#include<iostream>
using namespace std;
typedef long long LL;  //long long存数据 typedef 类型别名
LL gcd(LL a,LL b) //gcd模板
{
   if(a<b)return gcd(b,a);
   if(!b)return a;
   return gcd(b,a%b);
}
LL lcm(LL a,LL b)
{
   return a*b/gcd(a,b); //保证可以整除,省略证明
}
```

```
int main()
{
    int n;
    while(cin>>n)
    {
        LL ans;
        cin>>ans;
        while(--n)
        {
            int t;
            cin>>t;
            ans=lcm(ans,t);
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

G - pipes

题意:

首先输入一个q为询问的组数,然后输入n,为两排水管每排有n个管子, 接下来两串长度为n的字符串表示每排水管的形状,要求输出是否可以 通过旋转水管来使从左上角水流通到右下角.水管共有6种形状, 具体可以看原题的图片理解.

样例输入

6

7

2323216

1615124

1

3

4

2

13

24

2

12

34

3

536

345

2

46

54

3 2 5 7

样例输出

YES

YES

YES

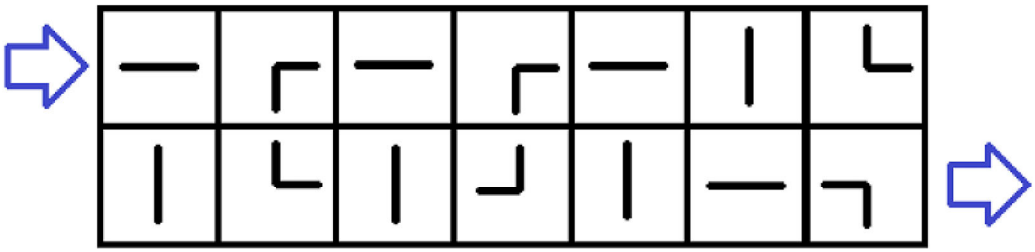
NO

YES

NO

思路

通过题目描述,我们可以知道1和2两种形状的水管可以相互转化, 而3-6的水管也可以相互转化,那么我们直接把水管分成这两大类即可. 观察可以得知,第一类型水管只能横着接第一类或者第二类, 而第二类水管则只能和第二类水管竖着相接以及和第一类横着接. 那么分析可以得知,只有当第二类水管竖着不能接第二类的时候不可以成立, 以及最终无法到达右下角的终点时候.
最后我们的做法就是从左上角模拟一遍水流的流向看是否能流到右下角.

> 分析第一个样例
>
> 
>
> 刚开始是碰到一个第一类 -> 只能同排走 -> 碰到一个第二类 -> 只能换行 -> 也是一个第二类即可行 -> 继续走是第一类不管 -> 继续碰到第二类 -> 只能换行 -> 依然是第二类为可行 -> 再走两个均为第一类可行 -> 碰到第二类 -> 换行查看也是第二类为可行 ->到终点 ->最终可行

代码

```
#include<iostream>
#include<string> //C++中的字符串类型头文件
using namespace std;
int main()
{
   int q;
   cin >> q;  //输入组数
   while (q--)
   {
       int n;
       cin >> n;
```

```
        string line1, line2;  //两排水管
        cin >> line1 >> line2;
        int flag = 0;  //一个标志着当前是模拟到哪一排
        for (int i = 0; i < n; i++) { //前进着模拟
            if (!flag) {   //如果是第一排的话
                if (line1[i] > '2')   //如果是第二类
                {
                    if (line2[i] <= '2') {   //但是换行之后的不是第二类就肯定不行
                        cout << "NO" << endl;
                        goto out;  //跳出循环
                    }
                    else flag = 1;  //可行的话就记录现在为第二行
                }
            }
            else {  //如果是第二排,同理第一排
                if (line2[i] > '2')
                {
                    if (line1[i] <= '2') {
                        cout << "NO" << endl;
                        goto out;
                    }
                    else flag = 0;
                }
            }
        }
        if (flag)cout << "YES" << endl;  //如果最终走到的是第二排则可以
        else cout << "NO" << endl;
    out:
        ;
    }
    return 0;
}
```

J - 饭卡

题意:

如果饭卡上的当前钱数大于等于五块,那么就可以买任意价钱的菜, 即使扣完之后为负,如果小于五块,那么什么都不能买.

样例输入

1

50

5

10

1 2 3 2 1 1 2 3 2 1

50

0

样例输出

-45

思路

因为小于五块就什么都买不了,所以先预留五块出来,最后买. 那么最后买什么呢,当然是买最贵的那个东西,这样子你才能负得最多. 至于为什么呢,我们来简单证明一下:假如你现在有x+5元,菜钱为y元, 那么最后的钱数为x+5-y, 显然y越大,这个数就越小. 除此之外,我们还注意到了需要让x的价钱最小,那么什么时候x的钱数最小呢? 这里就要用到动态规划问题的做法(dynamic problem)来计算x的最小值. x的最小值即为总钱数-5-dp中当dp为最大的时候. 那么问题就转化为了求除了最贵的菜以外,如何用总钱数-5块的钱数可以买到 总价最高的菜.转化之后就直接套用一下01背包dp的思路就好,在此说不清楚, 可以自行百度学习或者集训时候学习.

成哥的代码

```
#include<bits/stdc++.h>  // 万能头文件 部分编译器不能用 包括了全部头文件 比赛求快可以用
typedef long long ll
using namespace std;

const int maxn=1010;
int val[maxn];
int dp[maxn];

int main(){
   int n,i,j,money;
   while(~scanf("%d",&n) && n){
       memset(val,0,sizeof(val));
       memset(dp,0,sizeof(dp));
       for(i=0;i<n;++i)
           scanf("%d",val+i);
       scanf("%d",&money);

       if(money<5){  // 低于5元直接输出
           printf("%d\n",money);
           continue;
       }

       sort(val,val+n);
       int Max=val[n-1];
       money-=5;  // 用5元来买最贵的东西

       if(money>=val[0]){  // 如果剩下的钱 连最便宜的物品都买不起，那就不进行操作
           for(i=0;i<n-1;++i){  // i<n-1 因为最贵的东西 已经买走
               for(j=money;j>=val[i];--j){
                   dp[j]=max(dp[j],dp[j-val[i]]+val[i]);  // 状态转移方程
               }
           }
       }
       printf("%d\n",money+5-dp[money]-Max);
   }
   return 0;
}
```