Problem A. 计算幂次

难度	考点
1	函数的调用

题目分析

直接调用函数输出结果即可。

示例程序

```
#include<stdio.h>
int qpow(int a,int b,int p)
{
    int ret=1;
    while(b){
        if(b&1) ret=1ll*ret*a%p;
        a=1ll*a*a%p,b>>=1;
    }
    return ret;
}
int main(){
    int a,b,p;
    scanf("%d%d%d",&a,&b,&p);
    printf("%d",qpow(a,b,p));
}
```

Problem B. 错误的斐波那契数列

难度	考点
1	递归

题目分析

参考一下课件中的斐波那契数列的示例程序，注意除了将递归公式修改为$f(n-3)+f(n-1)$之外，递归终止调节也需要修改为 $n \leq 3$ 时返回 1。

示例程序

```
#include<stdio.h>
int f(int n)
{
    if(n<=3)
        return 1;
    else
        return f(n-3)+f(n-1);
```

```
}
int main()
{
    int x;scanf("%d",&x);
    printf("%d\n",f(x));
    return 0;
}
```

Problem C. 输出距离

难度	考点
1	自定义函数的应用

题目分析

此题建议采用自定义函数的方式通过，否则代码量会比较大。相信你应该对模块化编程的好处略有感知了。

不过仍有需要注意的地方。`sqrt` 函数需要头文件 `math.h`，小心自定义函数里相减对象搞错。

来自加提示的人的道歉：其实不需要用六个参数，像提示那样写太复杂了*QAQ*。

参考程序

```
#include<math.h>
#include<stdio.h>

int x[5], y[5], z[5];

int sqr(int x);
double dist(int i, int j);

int main()
{
    for(int i = 1; i <= 4; i++)
        scanf("%d%d%d", &x[i], &y[i], &z[i]);

    printf("%.2lf\n%.2lf\n", dist(2, 4), dist(1, 3));
    printf("%.2lf\n%.2lf\n", dist(2, 3), dist(3, 4));
    printf("%.2lf\n%.2lf\n", dist(1, 2), dist(1, 4));

    return 0;
}

double dist(int i, int j)
{
    return sqrt(sqr(x[i]-x[j])+sqr(y[i]-y[j])+sqr(z[i]-z[j]));
}

int sqr(int x)
{
    return x*x;
}
```

Problem D. xf学组合数学

难度	考点
3	递归，递推

题目分析

可以使用高中学过的阶乘求法，也可以使用组合数递归关系递归求解，递归时注意写明边界条件。

示例代码

```
#include<stdio.h>
int n,m,t;
int c(int m,int n)
{
    if(n==0)
        return 1;
    if(n==m)
        return 1;
    return c(m-1,n-1)+c(m-1,n);
}
int main()
{
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&m,&n);
        printf("%d\n",c(m,n));
    }
    return 0;
}
```

Problem E. GPA的计算

难度	考点
2	循环

题目分析

按题目所给的公式进行计算即可，需要注意的易错点：

- 两个`int`类型进行计算只会得到`int`类型的答案，本题中在进行除法运算前把它们转换为`double`（可以通过强制转化或者`*1.0`的方式）。
- **60分以下GPA为0**，即使加粗了还是有同学没看到QAQ。
- 公式看清楚，不要抄错了！

示例代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
#define N 105
int h[N];double g[N];
int main()
{
    int n,sumh=0;
    double sumg=0;
    scanf("%d",&n);
    for(int i=1,x,h;i<=n;i++)
    {
        scanf("%d%d",&x,&h);
        double g=4-3.0*(100-x)*(100-x)/1600;
        sumh+=h;
        if(x>=60) sumg+=g*h;
    }
    printf("%.2f\n",sumg/sumh);
    return 0;
}
```

Problem F. 素数日期

难度	考点
3	枚举、函数调用

题目分析

从当前日期的八位数开始枚举（至99991231），对于枚举的到的每个数，用所给的函数判断是否是合法日期以及是否的素数。

注：枚举八数字判断其是否是合法日期，远远比枚举日期要简单。

示例代码

```
#include<stdio.h>
int qpow(int a,int b,int p)
{
    int ret=1;
    while(b)
    {
        if(b&1) ret=1ll*ret*a%p;
        a=1ll*a*a%p,b>>=1;
    }
    return ret;
}
int isprime(int n)
```

```
{
    if(n==2||n==3) return 1;
    if(!(n&1)) return 0;
    int m=n-1,a,tmp,ans,cnt=0;
    while(!(m&1)) m>>=1,cnt++;
    int rd=20011224,seed=998244353,seed2=20217371;
    for(int i=0;i<20;i++)
    {
        rd=rd*seed+seed2;
        if(rd<0) rd=-rd;
        a=rd%(n-1)+1;
        tmp=qpow(a,m,n);
        for(int j=0;j<cnt;j++)
        {
            ans=1ll*tmp*tmp%n;
            if(ans==1)
            {
                if(tmp!=1 && tmp!=n-1)
                    return 0;
                break ;
            }
            tmp=ans;
        }
        if(ans!=1) return 0;
    }
    return 1;
}
int day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
int isday(int y)
{
    int d=y%100;y/=100;
    int m=y%100;y/=100;
    if(m<1 || m>12) return 0;
    else if(d==29&&m==2)
        return (y%100&&y%4==0)||y%400==0;
    return d>=1 && d<=day[m];
}
int Zeller(int y,int m,int d)
{
    if(m==1 || m==2) y--,m+=12;
    int c=y/100;y%=100;
    return ((y+y/4+c/4-2*c+26*(m+1)/10+d-1)%7+7)%7;
}
void print(int x)
{
    switch(x)
    {
        case 0:puts("Sunday");break ;
        case 1:puts("Monday");break ;
        case 2:puts("Tuesday");break ;
        case 3:puts("Wednesday");break ;
        case 4:puts("Thursday");break ;
        case 5:puts("Friday");break ;
        case 6:puts("Saturday");break ;
    }
}
int main()
{
```

```
    int now;
    while(~scanf("%d",&now))
    {
        int cnt=0;
        for(int i=now+1;i<=99991231;i++)
            if(isday(i))
            {
                cnt++;
                if(isprime(i))
                {
                    printf("%d %d ",i,cnt);
                    int d=i%100;i/=100;
                    int m=i%100;i/=100;
                    print(Zeller(i,m,d));
                    cnt=-1;break ;
                }
            }
        if(cnt!=-1) puts("-1");
    }
    return 0;
}
```

Problem G. 分数加法

难度	考点
3	递归

题目分析

保证输入数字$<= 1\times10^{16}$，约分后，$|a|,|b|,|c|,|d|\ <= 1\times10^{7}$。
所以开long long，读入a,b,c,d以后再约分成最简形式的a,b,c,d。
再算$(a*d+b*c)/b*d$（不会爆long long），再约分。

注意：

如果你样例对了但是你wa了，有两种原因。

1.你的符号没处理好，出现了`3/-5`这种输出。

2.你的计算过程爆long long了。

有的同学判断最后结果＜0，是这么判断的——“分子×分母＜0”，分子和分母都在long long范围内，不保证乘起来不会爆long long，应该用（分子＜0且分母＞0或分子＞0且分母＜0）；

有的同学算a.b最小公倍数，算的是a×b/gcd（a，b），同样的道理，a×b会爆long long，应该改成a/gcd（a，b）×b，爆数据范围的题希望大家格外注意。

示例代码

```
#include <stdio.h>
long long a, b, c, d, x, y, temp, flag = 0;
long long gcd(long long n, long long m)
{
```

```
    if (n % m == 0)
        return m;
    else
        gcd(m, n % m);
}
int main()
{
    while (~scanf("%lld/%lld%lld/%lld", &a, &b, &c, &d))
    {
        //----对第一个分数约分----
        temp = gcd(a, b);
        a /= temp;
        b /= temp;
        //----对第二个分数约分----
        temp = gcd(c, d);
        c /= temp;
        d /= temp;
        //----计算结果&对结果约分----
        y = b * d;
        x = a * d + b * c;
        temp = gcd(x, y);
        x /= temp;
        y /= temp;
        //----处理符号，保证如果有负号，负号在最前面----
        flag = (x<0&&y>0||x>0&&y<0) ? 1 : 0;
        x = x > 0 && flag ? -x : x;
        y = y < 0 ? -y : y;
        //----如果你看不懂上面带?和:的这几句，请百度一下三目运算符----
        printf("%lld/%lld\n", x, y);
    }
    return 0;
}
```

示例代码2

其实对`gcd`函数的递归过程进行一些处理（之前讲过的mod运算的性质），将负数转变为正数，就能省去符号的讨论。

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
#include <stdio.h>
long long gcd(long long a,long long b)
{
    return b?gcd(b,(a%b+b)%b):a;
}
int main()
{
    long long a,b,c,d,t;
    while (~scanf("%lld/%lld %lld/%lld",&a,&b,&c,&d))
    {
        t=gcd(a,b),a/=t,b/=t;
        t=gcd(c,d),c/=t,d/=t;
        a=a*d+b*c,b=b*d;
```

```
        t=gcd(a,b),a/=t,b/=t;
        printf("%lld/%lld\n",a,b);
    }
    return 0;
}
```

Problem H. 可爱的数列

难度	考点
4	递归、位运算

题目分析

- 解法1：递归，分析此题中递归的要素，递归的表达式题目已经给出，终止条件即为不断二分，数列长度不断减小，直至长度为 1 时，根据这些要素进行递归模拟即可。
- 解法2：设 $x=f([a_1,a_2,\cdots,a_k])$，$y=f([a_{k+1},a_{k+2},\cdots,a_{2k}])$；
  则：$f([a_1,a_2,\cdots,a_{2k}])=(x+y)\&(x-y)$，将 01 带入得：

$$(0+0)\&(0-0)=0$$
$$(0+1)\&(0-1)=1$$
$$(1+0)\&(1-0)=1$$
$$(1+1)\&(1-1)=0$$

  即$f([a_1,a_2,\cdots,a_{2k}])=x\bigoplus y$，根据异或运算的性质：

$$f([a_1,a_2,\cdots,a_n])=a_1\bigoplus a_2\bigoplus\cdots\bigoplus a_n$$

示例代码1

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
#define N 1005
int a[N];
bool dfs(int l,int r)
{
    if(l==r) return a[l];
    int mid=(l+r)>>1;
    int x=dfs(l,mid),y=dfs(mid+1,r);
    return (x+y)&(x-y);
}
int main()
{
    int T,n;
    scanf("%d",&T);
    while(T--)
    {
```

```
        scanf("%d",&n);
        for(int i=1;i<=n;i++)
            scanf("%d",&a[i]);
        printf("%d\n",dfs(1,n));
    }
    return 0;
}
```

示例代码2

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
int main()
{
    int T,n,x,ans;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n),ans=0;
        while(n--)
            scanf("%d",&x),ans^=x;
        printf("%d\n",ans);
    }
    return 0;
}
```

Problem I. $Verilog$ 匹配题解

难度	考点
5	字符串匹配、有限状态机

题目分析

- 解法1：采取直接判断子串是否为"begin"和"end"的方式进行判断，注意数组越界问题！！！`i+4<l`和`i+2<l`均为防止数组越界的判断，注意逻辑运算具有短路规则，即在与运算中，当第一个条件`i+4<l`不满足时，后面的判断会不再继续进行下去，注意输出`no`的情况分别有在中途end的数量就大于begin和最终begin和end数量不一致或者begin和end其中有一个没有存在。
- 解法2：采用有限状态机的方式进行判读begin和end，这样做可以规避掉可能存在的数组越界的问题，关于有限状态机的知识大家可以在网络上进行搜索，这里助教采用的是Mealy型状态机。

示例程序1：

```
#include <stdio.h>
#include <string.h>

char s[1000];
int statement;//如果Statement为1说明已经不满足匹配条件。
```

```
int main() {
    int i,l;
    int num_begin,num_end;
    while(gets(s)!=NULL) {
        num_begin = 0;//多组数据时每次循环的开头都要初始化变量。
        num_end = 0;
        l = strlen(s);
        statement = 0;
        for(i=0; i<l; i++) {
            if(statement==1) break;

if(i+4<l&&s[i]=='b'&&s[i+1]=='e'&&s[i+2]=='g'&&s[i+3]=='i'&&s[i+4]=='n') {
                num_begin++;
            }
            if(i+2<l&&s[i]=='e'&&s[i+1]=='n'&&s[i+2]=='d') {
                num_end++;
                if(num_end>num_begin) {//如果end的数量大于begin说明出现了嵌套错误，此
end没有对应的begin。
                    statement = 1;
                }
            }
        }

if(statement==1||num_begin!=num_end||num_begin==0||num_end==0)printf("no\n");
        else printf("yes\n");
    }
    return 0;
}
```

示例程序2：

```
#include <stdio.h>
#include <string.h>

char s[1000];
int statement;
int main()
{
    int i,l;
    int num_begin,num_end;
    while(gets(s)!=NULL)
    {
        num_begin = 0;
        num_end = 0;
        l = strlen(s);
        statement = 0;
        for(i=0;i<l;i++)
        {
            if(statement==7) break;
            else if(s[i]=='b')statement = 1;
            else if(s[i]=='e'&&statement==1)statement = 2;
            else if(s[i]=='g'&&statement==2)statement = 3;
            else if(s[i]=='i'&&statement==3)statement = 4;
            else if(s[i]=='n'&&statement==4)
            {
                statement = 0;
                num_begin++;
```

```
            }
            else if(s[i]=='e'&&statement!=1)statement = 20;
            else if(s[i]=='n'&&(statement==2||statement==20))statement = 5;
            else if(s[i]=='d'&&statement==5)
            {
                statement = 0;
                num_end++;
                if(num_end>num_begin)
                {
                    statement = 7;
                }
            }
            else statement = 0;
        }

 if(statement==7||num_begin!=num_end||num_begin==0||num_end==0)printf("no\n");
        else printf("yes\n");
    }
    return 0;
}
```

Problem J. 异或

难度	考点
5	位运算、递归

题目分析

既然是递归函数专场，那还是讲一讲递归的思路吧QAQ：

从高位往低位考虑，若某一位取 1 也满足 $\leq n$ 的条件，则更低位可以任意取值，即更低位的所有情况均能满足。

若这一位只能取 0，则是这一位与更高位的值均已固定，是一个对于更低位的子问题，即可用递归进行处理。

示例程序

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
int Min(int a,int b){return a<b?a:b;}
int dfs(int n, int m, int x, int y)
{
    if(m == 0) return n+1;
    else if(n < m) return 0;
    else if(!((n >> x) & 1))
        return dfs(n, m, x-1, y);
    else if(!((m >> y) & 1))
        return dfs(n, m, x, y-1);

    if(x > y)
```

```
    {
        int t = dfs((1<<x)-1, m, x-1, y);
        if(t < (1<<x))
            return t;
        t = dfs(n^(1<<x), m, x-1, y);
        if(t < (1<<x))
            return 1<<x | t;
        return 1<<(x+1);
    }
    else
    {
        if(n == (1<<(y+1))-1)
            return 1<<(y+1);
        int t = dfs(n^(1<<x), m^(1<<y), x-1, y-1);
        if(t < (1<<x))
            return t;
        else return 1<<(x+1);
    }
}
int main()
{
    int m, n;
    while(~scanf("%d",&m))
    {
        scanf("%d",&n);
        printf("%d\n",dfs(n, m, 30, 30));
    }
}
```