C7-2021级航类-第七次上机题解

A 简单的OI启蒙

难度	考点
1	循环、判断

题目分析

按照题目的意思模拟就可以了，只要苹果的高度**小于等于**陶陶的高度+凳子的高度，这个苹果就能摘到。

定义数组$int\ a[15]$存储10个苹果的高度，再定义$double\ h$来存储陶陶伸手的最大高度，定义$int\ maxh = h * 100 + 30$来存储陶陶站上凳子后能够到达的最大高度（注意h以米为单位，我们把它转化成厘米再赋给maxh，这样maxh和a[]单位相同可以直接比较），定义$int\ sum$来存储答案，初始化为0.

之后比较$maxh$和每一个$a[]$的大小，$maxh >= a[]$就让$sum + 1$，最后sum就是答案。

示例代码

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

int a[15], maxh, sum;
double h;

int main(){
    int i;

    for(i=1; i<=10; ++i)
        scanf("%d", &a[i]);
    scanf("%lf", &h);

    maxh = h*100 + 30;
    for(i=1; i<=10; ++i)
        sum += (maxh>=a[i]);

    printf("%d", sum);
    return 0;
```

```
}
```

B 求前导零的个数

难度	考点
1	位运算

题目分析

对于正整数从0到31位依次遍历,找非0的最高位即可
对于0直接输出

参考代码1

```c
#include <stdio.h>
int myclz(unsigned int x){
    int i = 0, hi = 0;
    if(x==0){
        return 32;
    }
    for(i = 0; i < 32; i++){
        if(x & (1 << i))hi = i;
    }
    return 31 - hi;
}
int main(){
    unsigned int x;
    scanf("%u", &x);
    printf("%d\n", myclz(x));
    return 0;
}
```

参考代码2

```c
#include <stdio.h>
int main(){

    unsigned int a,x, count=0;
    int i=0;
    scanf("%u",&a);

    for(i=31;i>=0;i--){
```

```
        x = a>>i;
        if( (x&1) == 0) count++;
        else break;
    }
    printf("%u\n",count);

    return 0;
}
```

C 简单结构体-给线段排排序

难度	考点
2	结构体，冒泡排序

题目分析

本题是最基础的结构体类型题目，要求大家在结构体的基础上实现简单的冒泡排序。显然本题用两个一维数组也能做，但这不是出题人的初衷，通过本题希望大家学会结构体的定义以及使用。

具体来说，我们可以定义一个结构体类型来表示每一条线段，其成员为两个int型变量，分别表示这条线段的左右端点，如下：

```
struct section {
    int l, r;//左端点和右端点
};
struct section temp;//声明一个结构体变量
struct section a[1010];//声明一个结构体数组
```

由此，我们定义一个结构体数组之后，按照冒泡排序的写法将n个结构体进行排序，唯一需要改变的是冒泡排序中交换的条件，具体见代码；同时在交换两个结构体变量时，按照类似int的交换方式即可，即：

```
//假设需要交换section结构体变量a和b
struct section temp=a;
a=b;
b=temp;
```

参考代码1

冒泡排序

```
#include <stdio.h>
#include <stdlib.h>
struct section
{
    int l,r;
};
struct section arr[1005];
int n;
int main(void)
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    scanf("%d%d",&arr[i].l,&arr[i].r);

    for(int i=0;i<n-1;i++)
    {
        int flag=1;
        for(int j=0;j<n-i-1;j++)
        {
            // 注意此处交换的条件
            if(arr[j].l>arr[j+1].l || (arr[j].l==arr[j+1].l &&
arr[j].r>arr[j+1].r))
            {
                struct section temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
                flag=0;
            }
        }
        if(flag)
        break;
    }
    for(int i=0;i<n;i++)
    printf("%d %d\n",arr[i].l,arr[i].r);

    return 0;
}
```

参考代码2

qsort快速排序

```
#include <stdio.h>
#include <stdlib.h>
// typedef的写法
typedef struct _section
{
    int l,r;
}section;
section arr[1005];
int n;
int cmp(const void *p1,const void *p2)
{
    section *a1=(section *)p1;
    section *a2=(section *)p2;
    if(a1->l < a2->l)
    return -1;
    else if(a1->l > a2->l)
    return 1;
    else
    return (a1->r < a2->r)?-1:1;
}
int main(void)
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    scanf("%d%d",&arr[i].l,&arr[i].r);

    qsort(arr,n,sizeof(arr[0]),cmp);

    for(int i=0;i<n;i++)
    printf("%d %d\n",arr[i].l,arr[i].r);

    return 0;
}
```

参考代码3

计数排序

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

```
int a[105][105];
int n,l,r;
int main(void)
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d%d",&l,&r);
        a[l][r]++;
    }
    for(int i=0;i<=100;i++)
    {
        for(int j=0;j<=100;j++)
        {
            for(int k=0;k<a[i][j];k++)
            printf("%d %d\n",i,j);
        }
    }

    return 0;
}
```

D 矩形重叠面积

难度	考点
2	结构体，判断

题目分析

找到重叠部分的左右上下边界即可，如左边界是两矩形左下角坐标的较大者。注意两个矩形不相交的情况，可额外对边界进行判断，或是采用参考代码中右边界选取“预设”右边界和左边界较大值的方法，若不相交，左边界和右边界取值相同，面积自然为0

还需注意虽然坐标值在int范围内，但面积有可能超出int，所以结果应使用$longlong$数据类型，部分解析见代码。

示例代码

```
#include <stdio.h>
#define max(a, b) (a > b ? a : b)
```

```
#define min(a, b) (a > b ? b : a)
struct pt_2d
{
    int x,y;
};
struct rect_2d
{
    struct pt_2d left_botoom,right_top;//结构体嵌套
};
int main()
{
    struct rect_2d rect1,rect2;
    int left,right,bottom,top;
    long long area;

scanf("%d%d%d%d",&rect1.left_botoom.x,&rect1.left_botoom.y,&rect1.
right_top.x,&rect1.right_top.y);

scanf("%d%d%d%d",&rect2.left_botoom.x,&rect2.left_botoom.y,&rect2.
right_top.x,&rect2.right_top.y);
    left=max(rect1.left_botoom.x,rect2.left_botoom.x);
    right=max(min(rect1.right_top.x,rect2.right_top.x),left);//右边
界设为 左边界 与 两矩形右上角横坐标较小者 中的较大值
                                                        //经过
此处运算，当left==right时，证明没有重叠部分，面积为0
    bottom=max(rect1.left_botoom.y,rect2.left_botoom.y);
    top=max(min(rect1.right_top.y,rect2.right_top.y),bottom);
    area=1LL*(right-left)*(top-bottom);//1LL代表longlong数据类型的1，
如果直接写1的话默认为int型的1
                                        //此处为使式子以longlong数据类
型进行运算所以使用1LL
    printf("%lld",area);
    return 0;
}
```

E 结构体小练习

难度	考点
2	结构体数组

题目解析

本题按题目要求完成每步操作即可。注意 11 位的手机号需要使用 `long long` 保存。

示例代码

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#include<ctype.h>

typedef struct Stu{
    char name[55];
    int id;
    char gender[10];
    long long phone;
}stu;//定义一个结构体

int main()
{
    stu info[105];//定义结构体数组
    int n,m,i,j,a,id;
    long long phone_new;
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++)
    {
        scanf("%s %d %s %lld",info[i].name, &info[i].id,
info[i].gender, &info[i].phone);
    }
    for(i=0;i<m;i++)
    {
        scanf("%d",&a);
        if(a==1)//在末尾添加一组数据
        {
            scanf("%s %d %s %lld",info[n].name, &info[n].id,
info[n].gender, &info[n].phone);
            n++;
        }
        else if(a==2)//修改联系方式
        {
            scanf("%d %lld",&id,&phone_new);
            for(j=0;j<n;j++)
            {
                if(info[j].id==id)//查找指定ID
                {
                    info[j].phone=phone_new;
```

```
                    break;
                }
            }
        }
    }
    for(i=0;i<n;i++)//按顺序输出
    {
        printf("%s %d %s %lld\n",info[i].name, info[i].id,
info[i].gender, info[i].phone);
    }
    return 0;
}
```

F 宋老师的名次预测5.0

难度	考点
4	结构体、快速排序

问题分析

经典结构体多关键字排序，定义一个结构体类型，含有每一位预测者的姓名、组别以及猜对的人数

```
typedef struct stu
{
    int group;      // 组别
    char name[30]; //姓名
    int guess;      //猜对的人数
} STU;
```

需要注意的是，本题排序逻辑较多，且数据量达到 10^5 ，因此冒泡排序等 $O(n^2)$ 复杂度的排序代码无法通过，推荐使用结构体+qsort快速排序，我们只需要在比较函数cmp中进行排序逻辑的书写即可

参考代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct stu
{
```

```
    int group;      //组别
    char name[30]; //姓名
    int guess;      //猜对的人数
} STU;
int cmp(const void *p1, const void *p2) // 重点在这
{
    const STU *a1 = (STU *)p1;
    const STU *a2 = (STU *)p2;

    if (a1->guess < a2->guess)
        return 1;
    else if (a1->guess > a2->guess)
        return -1; //首先根据猜对的人数返回
    else            //如果猜对人数相同
    {
        if (a1->group < a2->group)
            return -1;
        else if (a1->group > a2->group)
            return 1;                               //按照组别大小返回
        else                                        //如果组别也相同
            return strcmp(a1->name, a2->name); //按照姓名字典序大小返
回
    }
}
STU a[100005];
int n;
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d%s", &a[i].group, a[i].name); //读入数据，注意
a[i].name前面没有&
        int sum = 0, tmp;
        for (int j = 1; j <= 50; j++)
        {
            scanf("%d", &tmp);
            sum += (tmp == j); //计算猜对的人数
        }
        a[i].guess = sum;
    }

    qsort(a, n, sizeof(a[0]), cmp); //快速排序

    for (int i = 0; i < n; i++)
        printf("%s %d %d\n", a[i].name, a[i].group, a[i].guess);
//输出
```

```
    return 0;
}
```

G 图图玩转随机数

难度	考点
3	结构体排序

题目分析

由于“线性同余”算法是一个递推的序列，因此如果我们在 $X(0)$ 到 $X(10000)$ 之间观测到了一样的数字，就能够得出这个序列的周期 T，假设有 $X(i)=X(j)$（$i<j$ 且 $\forall\, i<k<j,\ X(i)\neq X(k)$），那么序列周期 $T=j-i$。

于是问题转变为判断数组 $X(0)$ 到 $X(10000)$ 中是否存在重复数字以及它们出现的位置，如提示所说，常用方法有两重循环查找、排序、利用数组下标，我们从这三种方法中选择合适的一种。

首先考虑两重循环查找，由于数组中一共有 10^4 个元素，且最多有 100 组数据，因此 $O(n^2)$ 的时间复杂度是无法通过的；再考虑先排序再查找，时间复杂度为 $O(nlogn)$，合适；最后考虑把 $X(i)$ 作为数组下标进行计数，虽然时间复杂度为 $O(n)$，但是考虑到数组中的元素 $X(i)$ 最大可达 2^{32}，而我们没法开出一个具有 2^{32} 个元素的数组，于是这种方法也不适合。

于是确定解题方法为排序，且要对数组的下标进行“捆绑”排序，容易想到我定义一个结构体 `Item`，里面放着一个数字 `num` 和这个数字在数组中的位置 `pos`，我们对结构体进行多关键字排序，在 `num` 不相等的时候按照 `num` 升序排列（或降序），在 `num` 相等的时候按照 `pos` 升序排列（或降序），排完序之后进行查找，如果存在紧挨着的两个 `num` 成员相等的对象，那么它们的 `pos` 之差即为序列的周期 T。

示例代码

```
#include <stdio.h>
#include <stdlib.h>
#define ONEW 10000      // 宏定义，1万

typedef struct {
    long long num;
    int pos;
} Item;     // 定义一个结构体Item，具有两个成员：数字num和其下标pos
```

```c
int r_cmp(const void *p1, const void *p2) {
    Item *pp1 = (Item*)p1;
    Item *pp2 = (Item*)p2;
    if (pp1->num > pp2->num)
        return 1;
    else if (pp1->num < pp2->num)
        return -1;
    else
        return pp1->pos - pp2->pos;      // 在num相等的情况下，按照pos的大小进行排序
}

int main() {

    int a, c, seed, i, T;
    long long m;          // m要定义为long long
    Item x[10005];  // 定义Item类型的数组变量x
    while (scanf("%d%d%lld%d", &a, &c, &m, &seed) != EOF) {
        x[0].num = seed;
        x[0].pos = 0;
        for (i = 1; i <= ONEW; i++) {
            x[i].num = (a * x[i - 1].num + c) % m;  // 线性同余公式
            x[i].pos = i;         // 下标i
        }

        qsort(x, ONEW + 1, sizeof(x[0]), r_cmp);     // 注意数据总数为一万加一

        T = -1;
        for (i = 0; i < ONEW; i++) {
            if (x[i].num == x[i + 1].num) {
                T = x[i + 1].pos - x[i].pos;    // pos成员直接相减得到周期T
                break;
            }
        }
        if (T == -1)
            printf("T > 10^4\n");
        else
            printf("T = %d\n", T);
    }

    return 0;
}
```

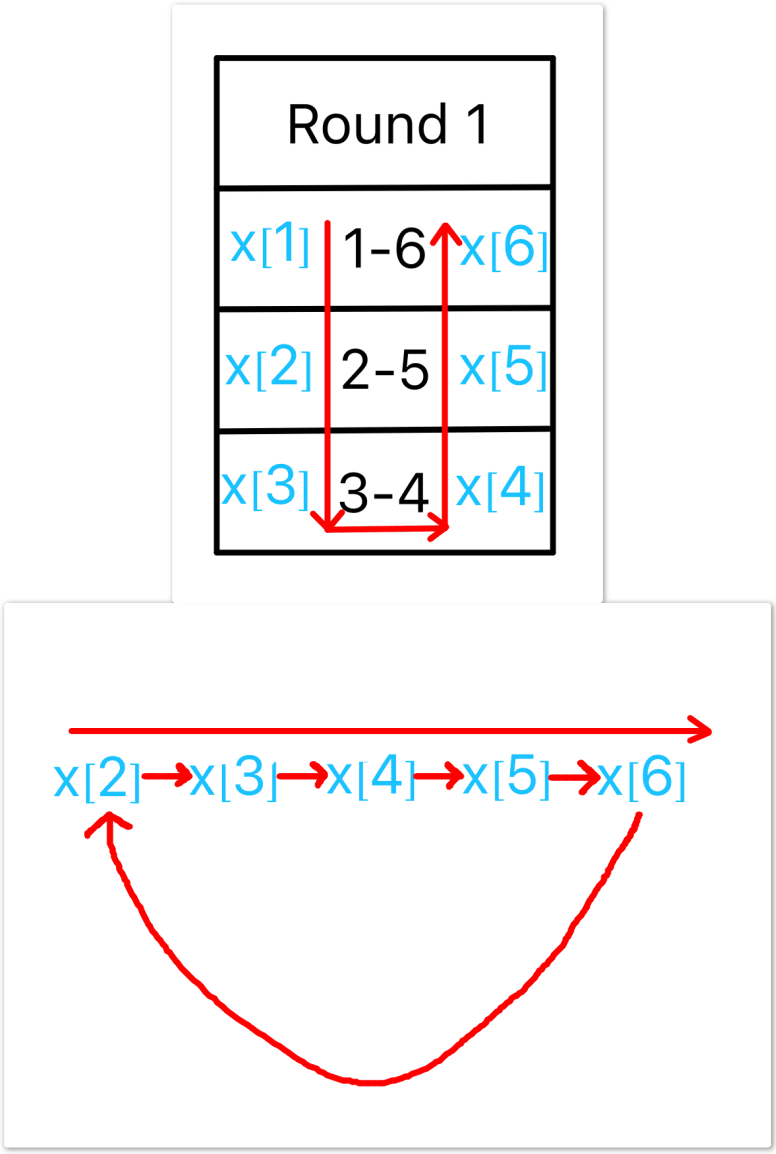
H 图图安排比赛

难度	考点
3	数组

题目分析

不要被长长的题面和逆时针旋转给吓唬到，其实选对方法后，我觉得比上一题要简单很多。

看到表格我们首先想到的是用二维数组来表示，但其实我们仔细观察，每逆时针旋转一次，相当于数组的一次循环移位，因此这道题用一维数组会比二维数组方便很多。



以 $n = 6$ 为例，一维数组 x 的编排顺序为第一列从上到下、第二列从下到上。在轮转时 $x[1]$ 固定不动，$x[2]$ 到 $x[6]$ 之间进行一次循环移位（如上图所示），即能达到“逆时针轮转”的效果。

这两道题提示我们，面对一个之前没见过的问题，须冷静下来分析，寻找突破口，再找对合适方法，能够事半功倍地解决问题。

示例代码

```c
#include <stdio.h>

int x[50];

void print(int n, int row) {

    for (int i = 1; i <= row; i++) {
        if (x[i] != 0 && x[2 * row + 1 - i] != 0) {      // 轮空不打印
            printf("%d-%d\n", x[i], x[2 * row + 1 - i]);
        }
    }

    return;
}

void rotate(int n, int row) {

    int temp = x[2 * row], i;        // 保存右上角的值为temp
    for (i = 2 * row; i >= 3; i--) {
        x[i] = x[i - 1];             // 数组后移一位
    }
    x[2] = temp;                 // temp送x[2]

    return;
}

int main() {

    int n, row, round, i;    // row表示每轮比赛有多少行，round表示总共多少轮比赛
    scanf("%d", &n);
    row = (n + 1) / 2;  // 每一轮中有(n+1)/2行
    for (i = 1; i <= 2 * row; i++) {
        x[i] = i;        // x数组赋初值
    }

    if (n & 1) {         // n为奇数
        round = n;
        x[2 * row] = 0; // 右上角置0
    } else {
        round = n - 1;
    }

    for (i = 1; i <= round; i++) {
```

```
        printf("Round %d\n", i);
        print(n, row);       // 打印该轮比赛
        rotate(n, row);      // 逆时针旋转
    }

    return 0;
}
```

I 简单的网页生成

难度	考点
4	字符串

问题分析

本题是一个字符串模拟问题。需要应用到字符串的分割，查找，比较，细节部分可能有一点多。

首先是输入部分的处理，网页源代码用字符数组存储就行。变量的定义部分则需要将变量名和变量的值分开识别，并且存储在字符数组。注意变量值可以按照两个 `\"` 来识别它的起点和终点。

输入处理完之后，可以边替换边输出。对每一行字符，我们查找两个连续的 `{` ，然后识别后面的变量名，然后直接输出变量名对应的变量值，并且跳过后面的两个 `}` 符号。

总的来说，思路较为直接。

示例代码

```
#include<stdio.h>
#include<string.h>
char str[10];
char varname[305][305],varvalue[305][305];
char text[305][305];
char tmp[305];
int main() {
    int n,m;

    scanf("%d%d",&n,&m);
    gets(str); //读入换行符
    int i,j;
```

```
    for(i=0;i<n;i++) {
        gets(text[i]); //读入源代码
    }
    for(i=0;i<m;i++) {
        scanf("%s",varname[i]); //读入变量名
        char c;
        c = getchar();
        while(c != '\"') c=getchar(); //跳过变量名和变量值之间的空格，
在第一个"停下
        c = getchar();
        j = 0;
        while(c != '\"') {
            varvalue[i][j] = c; //读入变量值，注意识别末尾的"
            j++;c=getchar();
        }
        gets(str);
    }

    for(i=0;i<n;i++) {
        for(j=0;j<strlen(text[i]);j++) {
            if (j+1<strlen(text[i]) && text[i][j]=='{' && text[i]
[j+1]=='{') { //判断是否是变量标记的起点
                int k=j+3,idx=0,l;
                memset(tmp,0,sizeof(tmp));
                while(text[i][k]!=' '||(text[i][k+1]!='}' &&
text[i][k+2]!='}')) { //读入标记中的变量名字
                    tmp[idx] = text[i][k];
                    k++;idx++;
                }
                for(l=0;l<m;l++) { //查找这个变量名字对应的变量值
                    if (strcmp(varname[l],tmp)==0) {
                        printf("%s",varvalue[l]);
                        break;
                    }
                }
                j = k+2; //注意跳过后面的}}字符
            } else {
                printf("%c",text[i][j]); //如果不是变量标记的起点就直接
输出
            }
        }
        printf("\n");
    }

    return 0;
}
```

J 命运的波导

难度	考点
4	结构体

如果熟悉复数计算类型的题目，那么这个题的解法非常简单。
比如：对于复数a+bi，可以定义一个结构体类型，里面封装两个成员：a和b。这在Hint中已经给出了相关内容。
本题要封装

这样的数，并且实现相应的乘法运算。
注意取模会将输出范围限定在0到998244352之间，负数需要翻译成相应的正数。涉及乘法操作，乘法的结果大约是998244353的平方，数据应该在long long范围。

```
#include<stdio.h>

struct SGL
{
    long long S;
    long long G;
    long long L;
};

struct SGL Mult(struct SGL A,struct SGL B)
{
    struct SGL C;
    C.S=
((A.S*B.S+2*A.G*B.L+2*A.L*B.G)%998244353+998244353)%998244353;
    C.G=
((A.S*B.G+A.G*B.S+2*A.L*B.L)%998244353+998244353)%998244353;
    C.L=((A.S*B.L+A.L*B.S+A.G*B.G)%998244353+998244353)%998244353;
    return C;
}

struct SGL FastPower(struct SGL base,long long exponent)
{
    struct SGL power;
    power.S=1;
    power.G=0;
    power.L=0;
    for(;exponent!=0;exponent>>=1)
    {
```

```
        if((exponent&1)==1)
        {
            power=Mult(power,base);
        }
        base=Mult(base,base);
    }
    return power;
}

int main()
{
    long long n;
    scanf("%lld",&n);
    if(n==0)
    {
        printf("1 0 0\n");
        return 0;
    }
    struct SGL ans;
    if(n>0)
    {
        struct SGL power;
        power.S=1;
        power.G=1;
        power.L=1;
        ans=FastPower(power,n);
    }
    if(n<0)
    {
        n=-n;
        struct SGL power;
        power.S=998244352;
        power.G=1;
        power.L=0;
        ans=FastPower(power,n);
    }
    printf("%lld %lld %lld\n",ans.S,ans.G,ans.L);
}
```

K zhnの超级数列

难度	考点
3	矩阵快速幂

题解

首先我们拿到这道题，很容易就能看出来，10^{18}肯定不能直接通过线性$O(n)$的复杂度计算，那我们应该怎么算呢？

首先介绍第一个东西：**矩阵快速幂**

在前面泡泡蛙蛙那道题已经给了快速幂的模板，这里就不加以赘述，和普通的快速幂一样，不过需要你自己定义一个乘法，说的更通俗一点，你需要自己写一个乘法来实现矩阵的乘法。如果你做过（看过）我之前出的那道组合数学的题，我也给过类似快速幂的板子，简单来说，它可以在$O(logn)$的时间内求出一个数的n次方。

那么这个东西和本题有什么关系呢？

~~这么大的数明显需要一个这样的时间复杂度的算法好吧~~

斐波那契数列的定义大家应该都很熟悉，不熟悉我题里也给了，我们定义如下的东西：

$Fib(n)$表示一个$1*2$的矩阵$[F_n\ F_{n-1}]$，我们希望通过矩阵运算的方式由$Fib(n-1)$来推出他，怎么推呢？

我们观察两个矩阵，是由$[F_{n-1}\ F_{n-2}]$ 乘以一个$2*2$的矩阵来得到$[F_n\ F_{n-1}]$

~~（如果你不知道为什么是2*2，魔法少女建议你重修线性代数）~~

结合斐波那契数列的通项公式，我们可以很容易地出来这个矩阵为$\begin{vmatrix}1 & 1\\1 & 0\end{vmatrix}$

所以最后的答案就是$[F_2\ F_1]*\begin{vmatrix}1 & 1\\1 & 0\end{vmatrix}^{n-2}$

可能有的小朋友会疑惑为什么是$n-2$次方，因为第一项和第二项不需要算，在矩阵里面有了，所以你手算一下，就知道为什么了。

如果还不明白，请返回上面再次大声朗读$Fib(n)$的定义。

更具体怎么写，在下面的代码中也给出了注释。

```
#include <stdio.h>
#define int long long
const int modd=1e9+7;
int n;
/*这里定义矩阵类型的结构体*/
typedef struct{
    int a[5][5];
}jv;
jv p,ans;
```

```
/*
  我定义了一个 jv类型的函数，他返回的类型，就是我上面定义的结构体
  这个函数就是传入两个矩阵，返回他们的乘积，中间的计算就是矩阵乘法
  为什么是2*2，因为我的快速幂是用来计算我构造的那个2*2的矩阵幂次的
*/
jv cheng(jv xx,jv yy){
     int sum;
     jv an;
     for(int i=1;i<=2;i++){
         for(int w=1;w<=2;w++){
             sum=0;
              for(int j=1;j<=2;j++){
                sum=(sum+xx.a[i][j]*yy.a[j][w]%modd)%modd;
              }
              an.a[i][w]=sum%modd;
         }
     }
     return an;

}

/*
    魔法少女的快速幂（矩阵版）
*/
jv pow(int x){
    jv temp=p;
    jv base=p;
    x--;
    while(x){
        if(x&1) temp=cheng(temp,base);
        base=cheng(base,base);
        x>>=1;
    }
    return temp;
}
/*因为我懒，把int定义成Long long了，所以这里不能写int main了*/
signed main(){
    scanf("%lld",&n);
    if(n<=2) printf("1");
    else{
        ans.a[1][1]=1;
        ans.a[1][2]=1;
        p.a[1][1]=1;
        p.a[1][2]=1;
        p.a[2][1]=1;
        p.a[2][2]=0;
        jv tt=pow(n-2);
        ans=cheng(ans,pow(n-2));
```

```
        int kk=(ans.a[1][1]%modd+modd)%modd;
        printf("%lld",kk);
    }
   return 0;
}
```

矩阵快速幂加速递推大概不在我们考试范围内，但可以当作一个有意思的idea欣赏一下，嘿嘿嘿