上机4解析

A 签到之GPA计算

难度	考点
1	数学计算

题目解析

签到题，按照题目要求进行计算即可

示例代码

```
#include<stdio.h>
int main()
{
    double sum = 0.0, GPA = 0.0, grade, num;
    int i, n;
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        scanf("%lf %lf", &grade, &num);
        GPA += grade < 60 ? 0 : num*(4.0-3.0*(100.0-grade)*(100.0-
grade)/1600.0);
        sum += num;
    }
    printf("%.4f\n", GPA / sum);
    return 0;
}
```

B 简单的三角形面积

难度	考点
2	数学计算

这是一道水水的签到题，给了三种提示对应三种做法，按照提示来做即可。

示例程序1

```c
#include <stdio.h>
#include <math.h>
int main()
{
    double x1,x2,x3,y1,y2,y3;
    double a, b, c, p, S;
    scanf("%lf%lf%lf%lf%lf%lf", &x1,&y1,&x2,&y2,&x3,&y3);
    a = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    b = sqrt((x3 - x1) * (x3 - x1) + (y3 - y1) * (y3 - y1));
    c = sqrt((x3 - x2) * (x3 - x2) + (y3 - y2) * (y3 - y2));
    p = (a + b + c) / 2.0;
    S = sqrt(p * (p - a) * (p - b) * (p - c));
    printf("%.4f", S);
    return 0;
}
```

示例程序2

```c
#include <stdio.h>
int main()
{
    double x1,x2,x3,y1,y2,y3;
    double v1x, v1y, v2x, v2y, S;
    scanf("%lf%lf%lf%lf%lf%lf", &x1,&y1,&x2,&y2,&x3,&y3);
    v1x = x2 - x1; v1y = y2 - y1;
    v2x = x3 - x1; v2y = y3 - y1;
    S = (v1x * v2y - v1y * v2x) / 2;
    printf("%.4f", S);
    return 0;
}
```

示例程序3

```c
#include <stdio.h>
int main()
{
    double x1,x2,x3,y1,y2,y3;
    double S;
    scanf("%lf%lf%lf%lf%lf%lf", &x1,&y1,&x2,&y2,&x3,&y3);
    S = (x1 * y2 + x2 * y3 + x3 * y1 - x1 * y3 - x2 * y1 - x3 * y2) / 2;
    printf("%.4f", S);
    return 0;
}
```

C SIR模型算感染人数

难度	考点
3	循环

题目分析

主要考察使用循环迭代求解，可以定义两个中间变量 `temp_1` 和 `temp_2`，然后更新 `S` 、 `I` 、 `R` 的值，最后四舍五入保留整数即可。

示例代码

```
#include<stdio.h>
#include<math.h>
int main()
{
    int i,n;
    double S,I,R,beta,gamma,temp_1,temp_2,N;
    scanf("%lf%lf%lf%d",&S,&I,&R,&n);
    scanf("%lf%lf",&beta,&gamma);
    N=S+I+R;
    for(i=0;i<n;i++)//循环迭代
    {
        temp_1=beta*S*I/N;
        temp_2=gamma*I;
        S=S-temp_1;
        I=I+temp_1-temp_2;
        R=R+temp_2;
    }
    printf("%.f\n",I);
    return 0;
}
```

D 求阿克曼函数

难度	考点
3	递归

题目分析

这个题比较简单，只要按找题目要求写出递归函数即可。由于这个函数的输出随 m 的增加增长迅速，因此我们只计算到 m 小于 3，n小于11$ 的情况。唯一要注意的就是递归中分三类判断。

示例代码

```c
#include <stdio.h>

int ack(int m, int n)
{
    if (m == 0)
    {
        return n + 1;
    }
    else if (n == 0)
    {
        return ack(m - 1, 1);
    }
    else
    {
        return ack(m - 1, ack(m, n - 1));
    }
}

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d", ack(a, b));
    return 0;
}
```

E 小明又去春游

难度	考点
3	组合数

题目分析

本题就是在算组合数 C_m^n，只不过 n 需要根据输入的星期数 x 转换一下，即 $n = 8 - x$。参考PPT的例 *a5-11*，稍加变换即可。

参考代码

```c
#include <stdio.h>

int fun(int m, int n) {
  if(m < n || m < 1 )
    return 0; // no solution
  if(m == n || n == 0)
    return 1; // only one solution
```

```
  if(n == 1)
    return m; // select one from total
  return fun(m - 1, n - 1) + fun(m - 1, n);
}

int main() {
  int n, m, x;
  scanf("%d%d", &m, &x);
  n=8-x;
  printf("%d", fun(m, n));
  return 0;
}
```

F Zeller 求星期

难度	考点
2	switch的使用 Zeller公式

题目分析

改自例a4-6，只是增加了处理1582年之前的公式和判断不存在的日期，总体比较简单，把PPT中的代码略加修改即可。

示例代码

```
#include<stdio.h>
int seek_w(int longday){
    int w,c,y,m,d;  //century,year,month,day
    y=longday/10000;    //先使y是四位数的形式 因为后边的y--可能导致借位
    m=(longday%10000)/100;
    d=longday%100;
    //Zeller公式
    if(m<=2) m+=12,y--;
    c=y/100;    //再处理y和c为Zeller公式要求的格式
    y=y%100;
    if(longday>=15821015){
        w=(y+y/4+c/4-2*c+(26*(m+1)/10)+d-1)%7;
        if(w<0) w+=7;
        return w;

    }
    else if(longday<=15821004){
        w=(y+y/4+c/4-2*c+(13*(m+1)/5)+d+2)%7;
        if(w<0) w+=7;
```

```
            return w;
        }
        return -1;
}
void printfWeek(int w){
        switch (w){
            case -1:
                printf("Nonexistent date!\n"); break;
            case 0:
                printf("Sun\n");break;
            case 1:
                printf("Mon\n");break;
            case 2:
                printf("Tue\n");break;
            case 3:
                printf("Wed\n");break;
            case 4:
                printf("Thu\n");break;
            case 5:
                printf("Fri\n");break;
            case 6:
                printf("Sat\n");break;
            default: break;
        }
}
int main(){
        int longday,w;  //longday:day的yyyymmdd形式
        while(~scanf("%d",&longday)){
            w=seek_w(longday);
            printfWeek(w);
        }
        return 0;
}
```

G这个勇者确实很菜所以过分慎重

难度	考点
3	斐波那契数列

题目分析

斐波那契数列的变形，考虑$n(n \geq 3)$的情况，设`f(n)`为对应的策略数，若第一天选修1，则总方案数为`f(n-1)`；若第一天选修2或3，则第二天必然选择与之对应的策略，其顺序可交换，两天内的方案共2种，剩余天数共`f(n-2)`种方案，故得到递推公式`f(n)=f(n-1)+2*f(n-2)`，初始值`f(1)=1`，`f(2)=3`（在不致引起混淆的情况下也可以认为`f(0)=1`），递推即可。

另外，此题也展现了递归的一大问题：若不进行数据存储管理，随着递归深度的增加，耗时将大大增加。此题中，若直接使用一般的递归函数计算，也可以得出正确答案。但是对子问题的处理将进行多次重复调用（如计算`f(n)`和`f(n-1)`时，都会计算`f(n-2)`）。实际操作中，往往采取数组等数据结构，将中间过程存储，当重复遇到某一问题，直接取出使用即可。

示例代码

```
#include<stdio.h>
int main()
{
    int f[255] = {0, 1, 3}, n, i;
    for(i = 3; i <= 250; i++)
    {
        f[i] = (f[i-1] + 2 * f[i-2]) % 1000007;
    }
    while(~scanf("%d", &n))
    {
        printf("%d\n", f[n]);
    }
}
```

H 甄医生找工作牌

难度	考点
5	循环，分支

题目分析

题目看似复杂，其实存在一定规律，所以希望同学们能够在列举出所有情况后归纳出最核心的一行公式，当然，即使不加以归纳，只是简单地罗列出所有的子情况也是可取的，只不过会增加时间成本和出bug的风险。
医生移动的规律是：无论他在哪一边，都可以使用a*b*c来表示移动方向。

示例代码

```
#include <stdio.h>

int main() {
    int n;
    int a, b, c, cur_id, i, cnt;
    int visited[2000] = {0}; //用来标记找过的床位
    scanf("%d %d", &cur_id, &n);
    visited[cur_id-1] = 1;
```

```
    while (1) {
        scanf("%d %d %d", &a, &b, &c);
        if (c == 0) {
            break;
        }
        if (a == -1) {
            cur_id = 2 * n + 1 - cur_id; //到对面
        }
        cur_id += c * a * b; //移动
        visited[cur_id-1] = 1;
    }
    for (i = 0, cnt = 0; i < 2 * n; i++) {
        if (visited[i]) {
            cnt++;
        }
    }
    printf("%d %d", cnt, cur_id);
    return 0;
}
```

I 置换的分解

难度	考点
2	数组，循环

题目分析

本题算法就是选取一个元，不断地求它变换后的结果，直到将它变换回自身，就得到一个包含这个元的轮换。

示例代码

```
#include <stdio.h>
int main()
{
    int num[101]={0},tran[101],i,n,a;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        scanf("%d",&a);
        tran[i]=a;//tran[]数组用于存变换
    }
    for(i=1;i<=n;i++)
    {
        if(num[i]==0&&tran[i]!=i)//num[]数组用于记录数字是否被遍历过了
```

```
        {
            a=i;
            do
            {
                printf("%d ",a);
                num[a]=1;
                a=tran[a];
            }while(a!=i);
            printf("\n");
        }
    }
    return 0;
}
```

J Four Pegs Hanoi

难度	考点
3	递归

题目分析

思路在题干的HINT里面写的很清楚了，只要注意x的取值范围为$1 \leq x < N$。

关于**数据范围**：由$F3(N) = 2 \times F3(N-1) + 1$及$F3(1) = 1$可知，$F3(N) = 2^N - 1$。又因为$F4(N) = \min\{2 \times F4(x) + F3(N-x)\}$，则当$N - x > 31$的时候，$F3(N-x)$就会超过int能表示的范围。

但是由于题目已经说到输出小于2×10^5，则当$F3(N-x) > 2 \times 10^5$，即$2^{N-x} - 1 > 2 \times 10^5$时，对应的$x$取值肯定不是最优解，故我们可直接跳过，不进行计算。

值得说明的是，示例代码中用了数组取记录递归调用的结果，可以减少递归调用的次数。这是一种优化，在一些题目中可以极大地降低运行时间，避免超时。

示例代码

```
#include<stdio.h>
#include<math.h>
int F3[105]={0,1,3},F4[105]={0,1,3};//使用全局变量的数组记录F3(N),F4(N)，并进行初始化

int hanoi3(int N) //三柱汉诺塔的步数
{
    return ((1<<N)-1); //由F3(N)=2*F3(N-1)+1及F3(1)=1可知，F3(N)=2^N-1

    //下面的注释内是常规的递归做法
```

```
    /*
    if (F3[N]!=0) return (F3[N]);
    F3[N]=2*hanoi3(N-1)+1;
    return (F3[N]);
    */
}

int hanoi4(int N) //四柱汉诺塔的步数
{
    unsigned int x,temp,sum=0xffffffff;     // 因为要选取最小值，所以初始化为一个较大的值
    if (F4[N]!=0) return (F4[N]); //用数组记录不同的N对应步数，可以减少递归调用的次数
    //0xffffffff是十六进制下的2^32，即unsigned int的最大值
    for (x=1;x<N;x++)
    {
        if (N-x>=20) continue;
        //由于2^10=1024，则2^20>10^6>2*10^5。所以当N-x>=20时，F3(N-x)=2^(N-x)-1肯定不是最优解
        temp=2*hanoi4(x)+hanoi3(N-x); //递推的公式
        if (temp<sum) sum=temp; //取最小值作为最终步数
    }
    F4[N]=sum;
    return (sum);
}

int main()
{
    int N;
    while (~scanf("%d",&N))
        printf("%d\n",hanoi4(N));
    return 0;
}
```

K水水の多项式加法

难度	考点
1	数组

题目分析

这波啊，这波是真水题。

直接使用`scanf`读入，用次数作为数组下标，对相应系数进行累加，使用神必代码进行输出。注意数组类型使用`long long`

示例代码

```
#include<stdio.h>
long long a[10007];
int main()
{
    long long c;
    char u;
    int p,i;
    while (~scanf("%lld%c%d", &c, &u, &p))
    {
        a[p] += c;
    }
    for (i = 0; i < 10007; i++)
    {
        if (a[i])
        {
            printf("%+lld%c%d", a[i], u, i);
        }
    }

    return 0;
}
```

L原根

难度	考点
3	函数接口调用、循环

题目解析

本题主要考察同学们调用接口完成程序的能力。

题目涉及到了一些新的概念（如欧拉函数），做题的时候可能会被迷惑。但题目提供了三个相关函数，我们只需要知道如何调用每个函数，就可以用简单的循环完成题目的任务。

实际代码编写中，这样的思想是很常见的：特定的功能交给标准库或第三方库的函数实现，我们只专注于主要部分的代码编写，而不关注每个功能内部具体的实现方式。

样例程序

```
#include <stdio.h>

int Phi(int x){
    int i;
    int ret = x;
    for(i = 2; i * i <= x; i++){
        if(x % i == 0){
```

```
            ret = ret / i * (i - 1);
            while(x % i == 0) x /= i;
        }
    }
    if(x > 1) ret = ret / x * (x - 1);
    return ret;
}

int GCD(int a, int b){
    return b ? GCD(b, a % b) : a;
}

int PowMod(int a, int t, int p){
    int ret = 1;
    while(t){
        if(t & 1) ret = ret * a % p;
        a = a * a % p;
        t >>= 1;
    }
    return ret;
}

int main(){
    int q;
    int p, phi, cnt;
    int g, i, flag;

    scanf("%d", &q);
    while(q--){
        scanf("%d", &p);
        phi = Phi(p);
        cnt = 0;
        for(g = 1; g <= p; g++){        // 条件 1：依次枚举 1 <= g <= p
            if(GCD(g, p) != 1)          // 条件 2：g 与 p 互质
                continue;
            flag = 1;
            for(i = 1; i < phi; i++)
                if(PowMod(g, i, p) == 1)        // 条件 3：对 1 <= i < phi, g ^
i % p == 1 都不成立
                    flag = 0;
            if(PowMod(g, phi, p) != 1)          // 条件 3：g ^ phi % p == 1 成立
                flag = 0;
            if(flag) printf("%d ", g);
            cnt += flag;
        }
        if(cnt == 0) printf("-1");
        printf("\n");
    }
```

```
    return 0;
}
```