Problem A. 连电路

难度	考点
1	循环、浮点数的计算

参考代码

```
#include<stdio.h>
int main(){
    double R,r;int op;
    scanf("%lf",&R);
    while(~scanf("%d%lf",&op,&r)){
        if(op==0) R+=r;
        else R=1/(1/R+1/r);
    }
    printf("%.2f\n",R);
    return 0;
}
```

Problem B. 小秋月打码

难度	考点
2	二维数组、格式化输出

题目分析

按题意模拟即可，需要注意的几个易错点：

- 边界的判断。
- 浮点数四舍五入的保留方法`(int)(1.0*sum/cnt+0.5)`。

参考代码

```
#include<stdio.h>
int a[105][105];
int d[8][2]={{-1,0},{1,0},{0,-1},{0,1},{-1,-1},{-1,1},{1,-1},{1,1}};
int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            scanf("%d",&a[i][j]);
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++){
            int sum=a[i][j],cnt=1;
            for(int k=0;k<8;k++){
```

```
                int x=i+d[k][0],y=j+d[k][1];
                if(x>=1 && x<=n && y>=1 && y<=m)
                    cnt++,sum+=a[x][y];
            }
            printf("%3d%c",(int)(1.0*sum/cnt+0.5),j==m?10:32);
        }
    return 0;
}
```

Problem C. 最长最短行

难度	考点
2	字符串、string.h函数、指针

题目分析

本题使用`#inlucde<string.h>`中的`strcpy`和`strlen`函数就可以轻松解决，也建议同学们参考ppt的最长行代码，尝试使用指针实现这个题目。

可以用于自测的特殊情况：

- 所有字符串长度相同。
- 存在长度为`0`的字符串。

参考代码

库函数版本：

```
#include<stdio.h>
#include<string.h>
char now[1005],longest[1005],shortest[1005];
int l=-1,s=10000,n;
int main()
{
    while(gets(now)!=NULL)
    {
        n=strlen(now);
        if(n>l)
        {
            l=n;
            strcpy(longest,now);
        }if(n<s)
        {
            s=n;
            strcpy(shortest,now);
        }
    }
    printf("%d\n%s\n%d\n%s",s,shortest,l,longest);
}
```

指针版本：

```c
#include<stdio.h>
#include<string.h>
#define N 2005
char str1[N],str2[N],str3[N];
int main(){
    char *now=str1,*longest=str2,*shortest=str3;
    int mxlen=0,mnlen=N;
    while(gets(now)!=NULL){
        int len=strlen(now);
        if(len>mxlen){
            mxlen=len;
            char *tep=now;
            now=longest,longest=tep;
        }
        if(len<mnlen){
            mnlen=len;
            char *tep=now;
            now=shortest,shortest=tep;
        }
    }
    //这一句比较重要，忽略的话所有字符串同一长度的情况会出现问题
    if(mnlen==mxlen) shortest=longest;
    printf("%d\n%s\n",mnlen,shortest);
    printf("%d\n%s\n",mxlen,longest);
}
```

Problem D. 位运算

难度	考点
3	位运算

题目分析

根据题目要求进行左移和右移运算的操作，如果绕不清楚的话进制转换后暴力交换也比较简单~

需要注意的地方有：

- unsigned long long；
- (1<<60)这样会溢出，应该(1ull<<60)；
- 注意位运算的优先级，记不清就打括号QAQ。

参考代码

位运算法：

```
#include<stdio.h>

#define ull unsigned long long

signed main(){
    ull n,m;
    while(~scanf("%llu%llu",&n,&m)){
        ull p=(1ull<<(m+1))-1,q=0;
        if(m<31) q=(n>>2*m+2)<<(2*m+2);
        printf("%llu\n",q|((n&p)<<(m+1))|((n>>(m+1))&p));
    }
}
```

暴力交换法：

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
bool b[100];
int main(){
    unsigned long long n;int m;
    while(~scanf("%llu%d",&n,&m)){
        for(int i=0;i<=63;i++)
            b[i]=((n>>i)&1)>0;
        for(int i=0,j=m+1;i<=m;i++,j++){
            bool t=b[i];b[i]=b[j],b[j]=t;
        }
        n=0;
        for(int i=0;i<=63;i++)if(b[i])
            n=n|(1ull<<i),b[i]=0;
        printf("%llu\n",n);
    }
}
```

Problem E. 点到谁就选谁

难度	考点
3	简单循环，判断，模拟

题目分析

简单模拟即可，具体可以参考代码。

参考代码

```
#include<stdio.h>

int a[100005];
int num[100005];
```

```
int main()
{
    int m, n, top = 0;
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &a[i], &num[i]);
    }
    for (int i = 0; i < m; i++) {
        int s, x;
        scanf("%d%d", &s, &x);
        if (s ^ a[top]) {    //利用异或确定传递方向， 1 为逆时针， 0 为顺时针
            top += x;
        }
        else {
            top -= x;
        }
        //接下来实现循环步骤
        if (top >= n) {      //若top大于等于n,则对top取模
            top %= n;
        }
        while (top < 0) {    //若top小于0，则循环对top加n，直到其为零或正数
            top += n;
        }
    }
    printf("%d", num[top]);
    return 0;
}
```

Problem F. xf当助教（一）

难度	考点
3	hash表

题目分析

首先看数据范围，10^5的范围不能用二重循环挨个计数，所以考虑有没有什么简化做法，aj-ai=j-i可以变换为aj-j=ai-i，即计算当前的值减去当前的数组下标的值相同的对数，开数组模拟哈希表即可，哈希表数组的下标为原始数组减去数组下标的值，注意数据范围，数组值减去数组下标可能出现负数，可以将哈希表数组下标都加上10^5避免出现负数

参考代码

```
#include<stdio.h>
int t,cnt[400010],a[400010],n;
long long ans;
int main()
{
    scanf("%d",&t);
    while(t--)
    {
        ans=0;
        //ans清零
```

```
        for(int i=0;i<=400000;i++)
            cnt[i]=0;
        //初始化，防止之前的计算结果对后续造成影响
        scanf("%d",&n);
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&a[i]);
            ans+=cnt[a[i]-i+100000];
            //数组值减去数组下标的哈希值加到ans中
            //整体数组下标+100000可以避免数组下标出现负数
            cnt[a[i]-i+100000]++;
        }
        printf("%lld\n",ans);
        //由于数据范围为10^5，ans可能超过int
    }
    return 0;
}
```

Problem G. 多关键字排序

难度	考点
4	排序，结构体，qsort的使用

解题思路

本题主要考察多个关键字的比较方法，可以按照常规的冒泡排序思路来做，循环对两个数的关键字进行比较，相等则`continue`，直到出现不等的情况，再根据当前关键字顺序的奇偶性（升序或降序）决定是否交换。

拓展：结构体+qsort

结构体貌似上课没讲，那么建议自学相关内容（挺简单的），在写qsort的比较函数cmp时注意类型的转换，以及权值在int范围内，要注意避免相减操作。下面是此种做法的示例代码。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
int n, k;
struct node{
    int id;
    int w[11];
}a[1010];
int cmp(const void *a, const void *b){
    struct node c = *(struct node *)a;
    struct node d = *(struct node *)b;
    int i;
    for(i = 0; i < k; ++i)
        if(c.w[i] == d.w[i]) continue;
        else return i % 2 ? (d.w[i] > c.w[i] ? 1 : -1) : (c.w[i] > d.w[i] ? 1 :
-1);
    return c.id - d.id;
```

```
}
int main(){
    scanf("%d%d", &n, &k);
    int i, j;
    for(i = 0; i < n; ++i){
        a[i].id = i + 1;
        for(j = 0; j < k; ++j)
            scanf("%d", &a[i].w[j]);
    }
    qsort(a, n, sizeof(a[0]), cmp);
    for(i = 0; i < n; ++i)
        printf("%d ", a[i].id);
    return 0;
}
```

Problem H. xf当助教（二）

难度	考点
4	思维

题目分析

数组元素有偶数个，xf和学长一人选择n/2个元素，由于xf会选择最优解，可以先算好数组中下标为奇数的元素和和下标为偶数的元素和哪个大，如果奇数元素和大就选择第一个，否则选最后一个，然后跟着学长的操作顺序即可确保拿到较大的元素和，所以先手必胜。

参考代码

```
#include<stdio.h>
int n,a[1000100];
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    printf("Ye$");
    return 0;
}
```

Problem I. 虚幻次方

难度	知识点
4	循环语句，条件判断

题目分析

本题主要考察大家对循环语句和条件判断的运用，在判断问题种类和输出时，都要进行比较复杂的条件判断。

关于化简根式$\sqrt{n}$的问题，即提取n最大的完全平方数因子，只需要从$\lfloor\sqrt{n}\rfloor$开始，往1依次枚举m，并判断完全平方数m^2是不是n的因子就行了。一旦找到了一个满足条件的m就可以退出循环了，因为如果n有比m^2小的完全平方数因子k^2，那么它一定还有一个比m^2更大的完全平方数因子$(mk)^2$，这与我们"m^2是第一个被找到的完全平方数因子"的假设是矛盾的。

关于i^k的问题，其实是对k以4为周期循环的。所以只需要取$k \div 4$的余数并进行判断就行了。

参考代码

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <stdlib.h>
#define ll long long
const double eps=1e-11;

void Task1(int x) { //sqrt(x)
    if(x==0) {
        printf("0\n");
        return;
    }

    int flag=0,a,b;
    if(x<0) {
        flag=1;
        x=-x;
    }

    int lim=sqrt(x)+1;
    for(int i=lim; i>=1; --i) {
        if(x%(i*i)==0) {
            a=i;
            b=x/(i*i);
            break;
        }
    }
    if(flag) {
        if(a==1 && b!=1)
            printf("sqrt(%d)*i\n",b);
        if(a!=1 && b==1)
            printf("%d*i\n",a);
        if(a==1 && b==1)
            printf("i\n");
        if(a!=1 && b!=1)
            printf("%d*sqrt(%d)*i\n",a,b);
    } else {
        if(a==1 && b!=1)
            printf("sqrt(%d)\n",b);
        if(a!=1 && b==1)
            printf("%d\n",a);
        if(a==1 && b==1)
            printf("1\n");
        if(a!=1 && b!=1)
```

```
            printf("%d*sqrt(%d)\n",a,b);
    }
}

void Task2(int x) { //i^x
    if(x%4==0) printf("1\n");
    if(x%4==1) printf("i\n");
    if(x%4==2) printf("-1\n");
    if(x%4==3) printf("-i\n");
}

int main() {
    int n;
    scanf("%d",&n);
    while(n--) {
        int o;
        scanf("%d",&o);
        if(o==1) {
            int k;
            scanf("%d",&k);
            Task1(k);
        } else if(o==2) {
            int k;
            scanf("%d",&k);
            Task2(k);
        }
    }
    return 0;
}
```

Problem J. 寻找单身狗

难度	考点
6	位运算

知识点分析

本题主要考察了位运算中的异或运算，其中，异或运算的真值表如下：

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

我们可以发现，对于一位二进制数字，两个相同的数字异或结果为0。推广到int型变量，我们能够得到：$a \oplus a = 0$, $a \oplus 0 = a$。同时，异或运算是具有交换律和结合律的。掌握了这些后，这道题就可以做啦。

题目分析

1. 对于没有单身狗的情况，只需要遍历一遍，维护最大值最小值，输出即可。
2. 对于一只单身狗的情况，同样只需要遍历一遍，把所有数字做异或操作，最后得到的答案就是结果。
3. 对于两只单身狗的情况，需要遍历两遍，第一遍遍历把所有数字做异或操作，得到的结果是 $x = a \oplus b$，因为 $a \neq b$，所以结果的二进制中至少有一位为 1，我们不妨找到最低位的 1，不妨设 a 的这一位为 1，b 的这一位为 0。我们再遍历一次数组，把这一位为 1 的所有数字做异或运算，得到的结果就是 a，此时 $b = a \oplus x$ $(x = a \oplus b)$。

注意：对于三种情况，要用 `if` 条件语句分清。第三种情况两个数字的乘积可能会超过 `int` 范围，请用 `long long` 输出。

参考代码

```
#include <stdio.h>
#define MaxInt 2147483647

int array[1000005];

int main()
{

    int ans = 0, max = 0, min = MaxInt, a, b; // 全局变量初始值为 0
    int N, temp, flag, position = 0, i, ans1;

    scanf("%d", &N);
    for (i = 0; i < N; i++)
    {
        scanf("%d", &array[i]);
        temp = array[i];
        ans = ans ^ temp;
        if (temp > max) max = temp;
        if (temp < min) min = temp;
    }
    if (ans == 0 && (N & 1) == 0)
    {
        printf("%d %d", min, max);
    }
    else if ((N & 1) == 1)
    {
        printf("1 %d", ans);
    }
    else
    {
        ans1 = ans;             // 将ans保存下来，以便后面求b
        while ((ans & 1) == 0) // 求a^b二进制中最低位为1的，则a和b对应的位置一定一个是1，另一个是0
        {
            ans = ans >> 1;
            position++;
        }
        for (i = 0; i < N; i++) // 重新遍历数组，只要该位是1，就进行异或运算，最后得到的是a
        {
            if ((array[i] >> position) & 1 == 1)
```

```
            {
                a = a ^ array[i];
            }
        }
        b = a ^ ans1; // 因为ans1 = a ^ b, 所以 b = an1 ^ a
        printf("2 %lld", (long long)a * (long long)b);
    }

    return 0;
}
```