Problem A. 一个简单的位运算

难度	考点
1	简单的循环、数组与位运算

题目分析

按位进行位运算并输出即可。

示例程序

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
#define N 105
int a[N],b[N];
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    for(int i=1;i<=n;i++)
        scanf("%d",&b[i]);
    for(int i=1;i<=n;i++)
        printf("%d",a[i]^b[i]);
    putchar(10);
    for(int i=1;i<=n;i++)
        printf("%d",a[i]|b[i]);
    putchar(10);
    for(int i=1;i<=n;i++)
        printf("%d",a[i]&b[i]);
    putchar(10);
    return 0;
}
```

Problem B. 又是一个简单的位运算

难度	考点
1	位运算的简单性质

题目分析

n个0或1进行位运算有以下性质：

- 异或：1的个数为奇数，结果为1，否则为0。
- 或：只有存在1，结果就为1。

- 与：全部为1，结果才为1。

示例程序

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
int main(){
    int n,x;
    while(~scanf("%d %d",&n,&x))
        printf("%d %d %d\n",x&1,x>0,x==n);
    return 0;
}
```

Problem C. MYX的函数

难度	考点
1	简单的循环与条件判断

题目分析

注意判断是否应该输出空格。

示例程序

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
int main(){
    int n;
    scanf("%d\n",&n);
    for(int i=1;i<=n;i++){
        char a,b;
        scanf("%c%c",&a,&b);
        if(a<b)
            printf("%c%c",a,i==n?10:32);
        else
            printf("%c%c",b,i==n?10:32);
    }
    return 0;
}
```

Problem D. 最大数字频数统计

难度	考点
2	循环，统计

题目分析

由题目条件，可以把第一个出现的数字当作最大数字，并统计为1。此后如果出现更大的数字则更新最大数字并将计数变量重新设置为1。

示例程序

```
#include<stdio.h>

int main()
{
    int cnt=1;
    int max_num;
    int n;
    int i;
    int a;
    scanf("%d",&n);
    scanf("%d",&max_num);
    for(i=1;i<n;i++)
    {
        scanf("%d",&a);
        if(a>n)
        {
            max_num = a;
            cnt = 1;
        }
        else if(a==n) {
            cnt++;
        }
    }
    printf("%d\n%d",max_num,cnt);
    return 0;
}
```

Problem E. Hamming Distance

难度	考点
2	位运算

题目分析

可以用异或运算将a，b中所有不同的位置1，相同的位置0。随后求出1的个数即可。

示例程序

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

```c
int main(void)
{
    unsigned long long a,b,c;
    int cnt = 0;
    scanf("%llu %llu",&a,&b);
    c = a^b;
    while(c){
        cnt+=c&1;
        c >>= 1;
    }
    printf("%d",cnt);
    return 0;
}
```

Problem F. 模拟汇编

难度	考点
3	进制转换、位运算

题目分析

- 方法一：模拟题目要求的过程，转二进制后提取对应位数转换即可。
- 方法二：使用位运算直接提取对应的二进制位。

示例程序1

```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
int a[35];
int main(){
    int n,x=0,y=0,z=0;
    scanf("%x",&n);
    for(int i=0;i<=30;i++)//转换为2进制，因为前6位永远为0，所以不用循环到32
        a[i]=(n>>i)&1;
    for(int i=7;i<=11;i++)//将前7-11位转换为10进制
        y=y<<1|a[32-i];
    for(int i=12;i<=16;i++)
        z=z<<1|a[32-i];
    for(int i=17;i<=21;i++)
        x=x<<1|a[32-i];
    printf("add $%d,$%d,$%d\n",x,y,z);
    return 0;
}
```

示例程序2

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
int main(){
    int n;
    scanf("%x",&n);
    printf("add $%d,$%d,$%d",(n>>11)&0x1F,(n>>21)&0x1F,(n>>16)&0x1F);
    return 0;
}
```

Problem G.士谔2173&&传源2173

难度	考点
3	标记、数组

题目分析

首先是如何判断一个数字是否为士谔传源数，采取每一位提取后计数的方式，较为简单，可以参考示例程序的代码。

本题的核心在于：可以用一个数组 $flag$ 标记数字是否为士谔传源数，如果是，则令`flag[i] = 1;`，否则，`flag[i] = 0;`。在每次输入 L 和 R 后循环遍历输出，如果该数字已经输出过了，则令`flag[i] = 0;`，这样就能够防止在每一次输入后都要判断 $[L,R]$ 范围内的数字是否为士谔传源数。

示例程序

```
#include<stdio.h>

int flag1[20002]; //全局数组不需要初始化，如果是局部数组，则需要初始化置0，memset函数或循环置0均可。
int main()
{
    int i,cnt,i1,i2;
    int L,R,n;
    scanf("%d",&n);
    for(i=1;i<=20000;i++)
    {
        i1 = i;
        cnt=0;
        while(i1!=0)
        {
            i2 = i1%10;
            i1/=10;
            if(i2 == 2 || i2 == 1 || i2 == 7)
            {
                cnt++;
            }
        }
        if(cnt==2||cnt==1||cnt==7)
        {
```

```
            flag1[i] = 1;//给士谔传源数进行标记
        }
    }
    while(n--)
    {
        scanf("%d%d",&L,&R);
        for(i=L;i<=R;i++)
        {
            if(flag1[i] == 1)
            {
                printf("%d ",i);
                flag1[i]=0;//给已经输出过的士谔传源数取消标记
            }
        }
        printf("\n");
    }
    return 0;
}
```

Problem H. 负二进制数

难度	考点
3	数学、进制

题目分析

将十进制整数转换为 m 进制数的方法是：将原数不断除以进制数 m，将得到的余数存入数组，一直到商为零时停止，最后逆序输出余数即可。

本题可能会出现余数为负数的情况，我们只需要将余数减去模数（除数），于此同时商再加上1，即可完成余数的转换，用余数公式来表示为：$被除数 \div 除数 = (商 + 1) \cdots\cdots (余数 - 除数)$。

注意 n 为 0 的情况。

示例程序

```
#include<stdio.h>
const int base = -2;
int main(){
    int t, n, i, temp, cnt;             //cnt表示位数
    int a[40];
    scanf("%d", &t);
    while(t--){
        cnt = 0;                        //初始化cnt为0
        scanf("%d", &n);
        if(n == 0){                     //n为0的情况
            printf("0\n");
            continue;                   //结束本次循环，然后进行下一次是否执行循环的判定
（与break不同）
        }
        while(n != 0){
            temp = n % base;
            if(temp < 0){               //余数为负
```

```
                a[cnt++] = temp - base; //余数减去模数
                n = n / base + 1;       //商加上一
            }
            else{                       //余数非负则不作调整
                a[cnt++] = temp;
                n /= base;
            }
        }
        for(i = cnt - 1; i >= 0; --i)   //余数倒序输出
            printf("%d", a[i]);
        printf("\n");
    }
    return 0;
}
```

Problem I. xf买彩票

题目分析

首先分析数字串中有没有出现过0或者8，如果出现过则一定是输出yes，然后不难想到，如果某个数能被8整除，则其要么是能被8整除的两位数，要么是结尾三个数构成的三位数能被8整除的多位数，那么相当于我们只需要找到数字串中是否存在一个可以被8整除的两位数或者三位数即可，注意三重循环时三个变量不能有重复。

```
#include<stdio.h>
#include<string.h>
int t;
char s[200];
int main()
{
    scanf("%d",&t);
    while(t--)
    {
        scanf("%s",s);
        int fl=0;
        int len=strlen(s);
        for(int i=0;i<len;i++)
        {
            if(s[i]=='0'||s[i]=='8')
            {
                printf("yes\n");
                fl=1;
                break;
            }
        }
        //存在8或者0，一定输出yes
        if(fl==1)
            continue;
        for(int i=0;i<len;i++)
        {
            for(int j=i+1;j<len;j++)
            {
                for(int k=j+1;k<len;k++)
                {
```

```
                    if(((s[i]-'0')*100+(s[j]-'0')*10+s[k]-'0')%8==0)
                    {
                        fl=1;
                        printf("yes\n");
                    }
                    //三重循环求出数字串中存在能被8整除的三位数，输出yes
                    if(fl==1)
                        break;
                }
                if(fl==1)
                    break;
                if(((s[i]-'0')*10+(s[j]-'0'))%8==0)
                {
                    fl=1;
                    printf("yes\n");
                    break;
                }
                //在与最内层循环平行的位置看有没有能被8整除的两位数，如果有则输出yes
                if(fl==1)
                    break;
            }
            if(fl==1)
                break;
        }
        //注意如果找到了多个符合要求的数，不能重复输出yes或者no，要利用好fl变量和break、
continue
        if(fl==0)
            printf("no\n");
        //一直没有找到符合要求的数，输出no
    }
    return 0;
}
```

Problem J. NAF

难度	考点
4	进制，数学

题目分析

本题由多种解法，不难发现二进制表示转化为NAF表示时，要将连续多位的“1”向高位推进。

不妨思考下以下两个等式：

1、$2^{n+1}+2^n=2^{n+2}-2^n$

2、$2^n+2^n=2^{n+1}$

本题解提供一个简单思路。

1、先将一个非负整数k转化成二进制数（利用数组存储每一位）。

2、从低位向高位遍历整个数组，遇到连续的两个“1”执行上述等式1操作，执行过程中遇到“2”执行上述等式2操作。

特别注意，要特判0。（以及最好每次读取一个非负整数时将数组清空）

示例程序

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

int main()
{
    int n, t, a[100] = { 0 };//注意初始化a[]
    long long c ;
    scanf("%d", &t);
    while (t--) {
        c = 1;
        scanf("%d", &n);
        memset(a, 0, sizeof(a));//养成好习惯，每次都把a数组重置一下
        for (int i = 0; i < 32; i++) {//利用位运算把整数的每一位存在数组里
            a[i] = (n & c) >> i;
            c <<= 1;
        }
        for (int i = 0; i < 32; i++) {//从低位向高位遍历，利用两个等式
            if (a[i] == 1 && a[i + 1] == 1) {
                a[i + 2]++;
                a[i + 1] = 0;
                a[i] = -1;
            }
            else if (a[i] == 2) {
                a[i + 1] ++;
                a[i] = 0;
            }
        }
        int flag = 0;//利用一个flag变量判断前导零
        for (int i = 32; i >= 0; i--) {
            if (a[i] == 1) {
                flag = 1;
            }
            if (flag) {
                printf("%d ", a[i]);
            }
            if (i == 0 && flag == 0) {//注意特判0
                printf("0");
            }
        }
        printf("\n");
    }
    return 0;
}
```