2021级航类-C2题解

A 水水的a+b>c

难度	考点
1	数据范围

题目分析

解答都在 HINT 中。若仅用 int 读入数据，那我们考虑两数相加溢出的情况，只可能是两个正数相加发生了溢出（此时 $a+b$ 必然大于 c），或是两个负数相加发生溢出（此时 $a+b$ 必然小于 c），据此我们可以写出示例代码 2，避免下次题目 a,b,c 范围改成在 $long\ long$ 以内。

示例代码1

```
#include <stdio.h>

int main()
{
    int n, i;
    long long a, b, c;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%lld%lld%lld", &a, &b, &c);
        if (a + b > c)
        {
            printf("true\n");
        }
        else
        {
            printf("false\n");
        }
    }
    return 0;
}
```

示例代码2

```
#include <stdio.h>

int main() {
    int n, a, b, c, i;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
```

```
    {
        scanf("%d%d%d", &a, &b, &c);
        int d = a + b;
        if (a > 0 && b > 0 && d < 0)
        {
            printf("true\n");
        }
        else if (a < 0 && b < 0 && d >= 0)
        {
            printf("false\n");
        }
        else if (a + b > c)
        {
            printf("true\n");
        }
        else
        {
            printf("false\n");
        }
    }
    return 0;
}
```

B 高斯求和

难度	考点
1	`for`循环

题目解析

给定等差数列的首项 a1 、公差 k 与项数 n 后，可以使用for语句或while语句对首项 a1 进行逐次累加，并将加和后的数字 ai 加和后存储到变量 sum 中，共累加n次。最后输出变量 sum 的值即可。

示例代码

```
#include <stdio.h>

int main(){
    int a, k, n, num;
    int i, sum;
    scanf("%d", &num);
    while(num--)
    {
        scanf("%d%d%d", &a, &k, &n);
        sum = a;//对sum的值初始化
        for(i = 1; i < n; i++)
        {
            a += k;
            sum += a;
        }
        printf("%d\n", sum);
    }
    return 0;
}
```

本题同时可以使用“等差数列求和公式”实现，代码如下：

示例代码2

```
#include <stdio.h>

int main()
{
    int a, k, n, sum, num;
    scanf("%d", &num);
    while (num--)
    {
        scanf("%d %d %d", &a, &k, &n);
        sum = (a + a + (n - 1) * k) * n / 2;
```

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

C 一元二次方程进阶版

难度	考点
2	复数的表示

题目解析

对每个方程的求解情况判断，注意严格按照输出格式进行输出。

示例代码

```c
/*
 Author: Songyou
*/

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

int main()
{
    int n, a, b, c, delta;
    double r1, r2;
    scanf("%d", &n);

    while (n--)
    {
        scanf("%d%d%d", &a, &b, &c);
        if (a == 0)
        {
            r1 = -c * 1.0 / b;
            printf("%.6f\n", r1);
        }
        else
        {
            delta = b * b - 4 * a * c;
            if (delta < 0)
            {
                //                printf("no real roots");
                double p = -b / (2. * a), q = sqrt(-delta) / (2 * a);
                if (q < 0)
                    q = -q;
                printf("%.6f+%.6fi %.6f-%.6fi\n", p, q, p, q);
            }
            else
            {
                r1 = (-b + sqrt(delta)) / (2 * a);
```

```
                r2 = (-b - sqrt(delta)) / (2 * a);
                if (r1 < r2)
                    printf("%.6f %.6f\n", r2, r1);
                else
                    printf("%.6f %.6f\n", r1, r2);
            }
        }
    }

    return 0;
}
```

D LJF的位运算加密

难度	考点
1	简单位运算

题目解析

将某一位上的数字置0可以通过与0进行与运算，若要保持其他位不变，则其他位需要与1进行与运算。

假设将一个8位二进制数的第1位置0，可以与`1111 1101`进行与运算，同时`1111 1101`可用(~(1 << 1))表示。

示例代码

```
#include <stdio.h>
int main()
{
    int a, x;
    scanf("%d", &a);
    for (int i = 0; i < 3; i++)
    {
        scanf("%d", &x);
        a &= (~(1 << x));
    }
    printf("%d", a);
}
```

E 统计字母

难度	考点
1	简单哈希表

题目解析

开一个长度26的数组arr存储A到Z的出现次数,对应关系为

- 大写字母ch的计数存到ch-'A'中
- 小写字母ch的计数存到ch-'a'中

示例代码

```
/*
 Author: Songyou
*/

#include <stdio.h>
#include <ctype.h>
#define N 26
int main()
{
    int i, c;
    int upper = 0, total = 0, alphabet[N] = {0};
    while ((c = getchar()) != EOF)
    {
        if (c >= 'a' && c <= 'z')
            alphabet[c - 'a']++; // if c is 'a', lower[0]++
        else if (c >= 'A' && c <= 'Z')
            alphabet[c - 'A']++; // if c is 'A', lower[0]++
    }
    for (i = 0; i < N; i++)
    {
        if (alphabet[i] != 0)
            printf("%c %d\n", i + 'A', alphabet[i]);
    }
    return 0;
}
```

F 小冰的烦恼

难度	考点
1	位运算

题目解析

本题的核心代码已经在提示中给出，首先判断第m和n位为0还是1，按照交换变量中采用临时变量存储的方法，并对相应位赋值。

先对hint中的代码进一步解释，以100十进制数字(二进制0110,0100)为例。

取出第二位：1<<2即0000,0100和0110,0100进行按位与运算，结果为0000,0100，大于0，说明100的二进制第二位为1，若结果为0，则二进制第二位为0。

第三位置1：1<<3即0000,1000和0110,0100进行按位与运算，结果为0110,1100，完成第三位置1。或运算的特点是不管是0还是1，和1或运算都为1。

第二位置0：1<<3即0000,0100，再取反得1111,1011和0110,0100进行按位与运算，结果为0110,0000，完成第二位置0。与运算的特点是不管是0还是1，和0与运算都为0。

本题中的1<<N,~1<<N的特殊变量通常被称作mask即遮罩，特点是只有特定位为0或1，其余位与之相反，在位运算中有着广泛的应用，大家在日后的学习中也会使用到它。

示例代码

```
#include<stdio.h>
int main()
{
    int x,m,n=0;
    scanf("%d %d %d",&x,&m,&n);
    int ms=x&(1<<m); //取出第m位
    int ns=x&(1<<n);//取出第n位
    if(ms)
    {
        x|=(1<<n); //第n位置1
    }
    else
    {
        x&=~(1<<n); //第n位置0
    }
    if(ns)
    {
        x|=(1<<m); //第m位置1
    }
    else
    {
        x&=~(1<<m); //第m位置0
```

```
    }
    printf("%d",x);
    return 0;
}
```

G 进制转换plus

难度	考点
2	进制

题目解析

本题主要考察进制转换的相关知识，10进制转换为k进制的基本思路是除k取余，倒序输出。本题需要注意的点是，对于a==0的情况，要进行特判。部分解析见代码。

参考代码

```
#include<stdio.h>
int main()
{
    int i,a,k;
    scanf("%d%d",&a,&k);
    int ans[101];//用于存储答案
    if(a==0)
    {
        printf("0\n");//输入为0的情况直接输出0，结束程序。
        return 0;
    }
    int num=0;
    while(a!=0)
    {
        ans[num]=a%k;//取余
        a/=k;//除k
        num++;//记录位数
    }
    for(i=num-1;i>=0;i--)
    {
        if(ans[i]<=9)
        {
            printf("%d",ans[i]);
        }
        else
        {
            printf("%c",ans[i]+55);//对于大于9的数字用大写字母代替
        }
    }
    return 0;
}
```

H 原码，反码，补码

难度	考点
3	二进制、位运算

问题分析

本题主要考察对于课件中原码，反码和补码相关知识的掌握情况，如果完全没有印象的话建议先好好复习一下课件中的内容，理解这三种编码的计算方法。

对于代码中提取数x二进制下第i位的代码：(x>>i)&1;

举个例子来帮助大家理解：

例如x=6，那么其二进制就为0000 0110，第0位为0，第1位为1，第2位为1，后面都为0。

- 第0位：x>>0等于x，即0000 0110，与0000 0001作&运算即为0；
- 第1位：x>>1等于0000 0011，与0000 0001作&运算即为1；
- 第2位：x>>2等于0000 0001，与0000 0001作&运算即为1；
- 后续均为0。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define ll long long
int a[10];
int main()
{
    int x;
    scanf("%d",&x);
    if(x>=0) //x大于0，原码、反码和补码相同
    {
        for(int i=0;i<8;i++)
        {
            a[i]=(x>>i)&1;
        }
        //将x的二进制表示存储到数组a中
        for(int t=1;t<=3;t++) //依次输出
        {
            for(int i=7;i>=0;i--)
            {
                printf("%d",a[i]);
            }
            printf("\n");
        }
```

```
    }
    else
    {
        x=-x;
        a[7]=1; //x<0，原码第一位符号位为1
        for(int i=0;i<7;i++)
        {
            a[i]=(x>>i)&1; //求绝对值的二进制
        }
        for(int i=7;i>=0;i--)
        {
            printf("%d",a[i]); //输出原码
        }
        printf("\n");
        for(int i=0;i<8;i++)
        {
            a[i]=!((x>>i)&1);
            /*
             * 求反码，为x的绝对值的二进制每位取反，可以用 ！逻辑运算符来帮助处理。
             * ！逻辑运算符与位运算中的取反运算符 ~ 不同，所得结果只能为0或1
             * !0=1 其余情况均等于0；~ 的话是将二进制下每一位取反，
             * 例如四位二进制数：0101 作 ~ 运算后变为:1010；作 ！运算后变为:0000
             * 若用 ~ 运算符的话，此处可以写成 a[i]=(((~x)>>i)&1);
             */
        }

        for(int i=7;i>=0;i--)
        {
            printf("%d",a[i]); //输出反码
        }
        printf("\n");
        a[0]++; //补码为反码+
        for(int i=0;a[i]>1;i++) //若+1后当前位为2，依次进位
        {
            a[i]=0;
            a[i+1]++;
        }
        for(int i=7;i>=0;i--)
        {
            printf("%d",a[i]); //输出补码
        }
        printf("\n");
    }
    // system("pause");
    return 0;
}
```

这份代码看起来显然不是很简洁，如果熟悉位运算的操作的话，可以尝试理解一下下面的代码：

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define ll long long
int main()
{
    signed char x;
    scanf("%hhd",&x);
    if(x>=0)
    {
        for(int k=1;k<=3;k++)
        {
            for(int i=7;i>=0;i--) printf("%d",(x>>i)&1);
            printf("\n");
        }
    }
    else
    {
        printf("1");
        for(int i=6;i>=0;i--)
        {
            printf("%d",((~(x-1))>>i)&1);
        }
        printf("\n");
        for(int i=7;i>=0;i--)
        {
            printf("%d",((x-1)>>i)&1);
        }
        printf("\n");
        for(int i=7;i>=0;i--)
        {
            printf("%d",(x>>i)&1);
        }
        printf("\n");
    }
    // system("pause");
    return 0;
}
```

送蚂蚁回家

难度	考点
3	思维

问题分析

本题是一个思维题，不需要按照题目要求模拟蚂蚁碰面掉头，这样是写不出来的；我们可以这样分析，假若任意两只蚂蚁碰面不掉头，一直按照原来的方向走，我们可以发现，整跟木棍上所有的蚂蚁的位置状态，跟题目所描述的情况，是一样的！也就是说，把两只蚂蚁碰面掉头看成穿透，这样来计算所有蚂蚁掉下木棍的时间。
因此，根据上述分析，我们只需要计算每一只蚂蚁“不掉头一直走”的掉落时间，然后取其中的最大值即可。

参考代码

```
#include <stdio.h>
#define MAX(a,b) ((a)>(b)?(a):(b))
int n,m;
int ans;
int main(void)
{
    scanf("%d%d",&n,&m);
    int x,a;
    for(int i=0;i<n;i++)
    {
        scanf("%d%d",&x,&a);
        if(a==1)
        {
            ans=MAX(ans,m-x); //如果是正方向，计算到末尾掉落时间取较大值
        }
        else
        {
            ans=MAX(ans,x); //如果是负方向，计算到木棍头掉落时间取较大值
        }
    }
    printf("%d\n",ans);
}
```

J 双重循环移位

难度	考点
3	位运算
### 问题分析	
如果将“循环移位”这个概念用在单独一个`int`类型变量上的话，可以想像是比较好实现的：	

```
int x = 100;
x = x << 1 | x >> 31; // 一次向左的循环移位
```

但是如果是本题定义的“双重循环移位”，是否还可以像上述代码一样的实现？答案是可以的，我们将两个`int`类型的变量“拼”成一个`long long`类型的变量即可，这样这道题目就变得简单了。
另外需要注意：对于一个`long long`类型的变量，由于它有64个`bit`，因此“循环移位”64次和“循环移位”0次是一样的，因此我们不必“循环移位”`n`次，“循环移位”`n % 64`次即可。

参考代码

```
#include <stdio.h>
typedef unsigned long long int ull;
typedef long long int ll;
int main(int argc, const char * argv[])
{
    ll a, b, mask = 0xffffffffLL; // LL 后缀用于代表long long类型的常量，比如1LL是long long类型的1
    int n;
    char op;
    while (~scanf("%lld %lld %c %d", &a, &b, &op, &n))
    {
        n %= 64; // 循环移位n % 64次
        ull ab = (a & mask) << 32 | (b & mask); // 将a和b“拼接”成unsigned long long型数据
        // 这里使用unsigned long long型数据是为了避免算数右移
        if (op == 'l')
        {
            ab = ab << n | ab >> (64 - n); //循环左移
        }
        else
        {
            ab = ab >> n | ab << (64 - n); // 循环右移
        }
        int na, nb; // 再将a和b“提取”出来
        na = ab >> 32 & mask;
        nb = ab & mask;
        printf("%d %d\n", na, nb);
    }
```

```
    return 0;
}
```