写在最前的tips(常见错误若干)

- **赋值** `=` 是赋值，`==` 才能判断是否相等
- **连等** `6<=x<=7` 在C语言中的意义与它在数学中的意义并不相当，`6<=x` 表达式的值真为1假为0，这样写会返回这个表达式的值与7比较的结果。正确的写法是 `6<=x&&x<=7`
- **格式** 注意输入输出格式，不要输出多余信息。如输出格式注明仅输出一个整数a+b，诸如 `printf("%d+%d=%d\n",a,b,a+b)` 的输出会被判为错误答案
- **还未开始就结束的if** 请不要在 `if` 后面直接跟 `;` （如果你不清楚这么做的意义的话）！！if后面如果跟大括号会执行括号内的内容，但如果没有的话仅会执行最近的一句话，当你直接跟 `;`，它什么都不会执行。

A. 为伟大的祖国母亲庆生吧！

解题思路

本题考察了对库函数 `printf` 的使用。 `printf` 可用于输出一字符串，只需将字符串用双引号包住即可。

注意，为了避免肉眼阅读失误，建议直接复制题中文本，而不是手打。

AC代码

```
#include <stdio.h>

int main()
{
    printf("I love you, China!");

    return 0;
}
```

B. 诶加闭

解题思路

本题考察了a+b和简单的输入输出。

在C语言中，我们可以使用 `scanf`、`printf` 函数进行数据的输入输出。需要注意每种数据类型对应的格式字符串，大家可以参阅书上的表格。

对于int，其格式字符串是%d。因此可以用scanf("%d%d", &a, &b);来读入int类型的a、b变量；同理可使用printf("%d", a+b);来输出a+b。

AC代码

```
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", a + b);

    return 0;
}
```

C. a*b problem

解题思路

本题考察了a*b和简单的输入输出。

与a+b类似，本题只需将a+b换成a*b即可。

AC代码

#include <stdio.h>

```
int main()
{
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", a * b);

    return 0;
}
```

D. 诶除闭

解题思路

这道题考察的是简单的输入输出，没有特别的要说的。

AC代码

```c
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%d%d", &a, &b);//读取一个数、字符的时候不要忘记加&，读取字符串时可以不加
    printf("%d", a / b);//整数除法会直接向下取整，故直接使用a / b即可
}
```

E. 简单a-b

解题思路

这道题考察的是简单的输入输出以及简单的分支，要注意由于由于需要判断的内容不止一个所以可以用嵌套的条件语句或者if(...&&...)来解决，思路分别是分层判断t和减数的值，另一种是将所有条件一同处理，对于复杂的问题，分层判断往往会更加清晰，下面给出两组代码。

AC代码

代码一：

```c
#include <stdio.h>
int main()
{
    int t, a, b;

    scanf("%d%d%d", &t, &a, &b);

    if(t == 1)//这个条件语句下面包含了两条要执行的内容必须使用大括号
    {
        if(b < 0) printf("%d-(%d)=%d", a, b, a - b);
//这个条件语句只有一个要运行的内容，可以直接在if()之后直接跟上要运行的内容
//但是在复杂的代码中，建议最好加大括号，代码更加清晰
        else printf("%d-%d=%d", a, b, a - b);
    }
    else
    {
        if(a < 0) printf("%d-(%d)=%d", b, a, b - a);
        else printf("%d-%d=%d", b, a, b - a);
    }
}
```

代码二：

```c
#include <stdio.h>
int main()
{
    int t, a, b;

    scanf("%d%d%d", &t, &a, &b);

    if(t == 1 && b < 0) {
```

```
        printf("%d-(%d)=%d", a, b, a - b);
    }
    else if(t == 1 && b >= 0){
        printf("%d-%d=%d", a, b, a - b);
    }
    else if(a < 0){
        printf("%d-(%d)=%d", b, a, b - a);
    }
    else{
        printf("%d-%d=%d", b, a, b - a);
    }
    //上面是一种可选的加大括号的方式，很多ide的自动格式化也会变成这个样子，但其他的易懂的标注形式也是可取的
    /*例如：
    if ()
    {
        ……
    }
    也是可取的标注办法*/
}
```

F. 转义字符

解题思路

考查知识点：转义字符。

思路：因为有些字符在printf 时可能会产生歧义，所以人们使用转义字符来输出这些字符，(请同学们熟悉一些常见的转义字符)，本题把需要转义的字符修改后，直接输出即可。代码如下：

AC代码

```
#include <stdio.h>

int main()
{
    printf("所有的转义字符和所对应的意义:\n"
    "字符    意义             ASCII码值（十进制）\n"
    "\\a      响铃(BEL)        007\n"
    "\\b      退格(BS)         008\n"
    "\\f      换页(FF)         012\n"
    "\\n      换行(LF)         010\n"
    "\\r      回车(CR)         013\n"
    "\\t      水平制表(HT)     009\n"
    "\\v      垂直制表(VT)     011\n"
    "\\\\      反斜线字符       092\n"
    "\\\'      单引号字符       039\n"
    "\\\"      双引号字符       034\n"
    "\\?      问号字符         063\n"
    "\\0      空字符(NUL)      000\n"
    "\\ddd    1到3位八进制数所代表的任意字符\n"
    "\\xhh    十六进制所代表的任意字符\n"
    "注意:\n1. 斜杠:\"/\"与反斜杠:\"\\\" ,此处不可互换\n"
    "2. \\xhh 十六进制转义不限制字符个数 \'\\x000000000000F\' == \'\\xF\'\n"
    "3. printf函数中\"%%\"作为格式控制符被占用,使用\"%%%%\"输出");
```

```
    return 0;
}
```

另外，在修改时，其实可以自己编写另外一个代码来生成输出数据，不过这个要进行文件操作，现在有点超纲，暂时不需要掌握，有兴趣的同学可以参考以下代码：

```
#include <stdio.h>

int main()
{
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);    //文件重定向，读取input文件，生成输出数据到output
    char c;
    while((c=getchar())!=-1)    //EOF=-1
    {
        if(c=='\\') printf("\\\\");
        else if(c=='\'') printf("\\\'");
        else if(c=='\"') printf("\\\"");
        else if(c=='%') printf("%%%%");
        else if(c=='\n') printf("\\n");
        else putchar(c);
    }

    return 0;
}
```

G. 宋老师的评语机器

解题思路

考察知识点：简单分支。

思路：根据输入的整数n，判断其所处的区间，然后输出相应的字符串。对于这种有多种选择的情况，我们可以使用if,else if语句来进行区间判断。另外，如果题目中要求输出一串字符串（特别是很长的）的话，建议大家可以直接复制粘贴到自己的代码中，这样不容易出错。

AC代码

```
#include <stdio.h>

int main()
{
    int n;
    scanf("%d",&n);
    if(n==10) printf("You are perfect! Please keep studying.");
    else if(n>=7) printf("You are great! Please  review and keep studying.");
    else if(n>=4) printf("You are good, but you need to review and listen more careful!");
    else printf("Please come to my office after class.");

    return 0;
}
```

H. 菲菲公主的ReLU (I)

解题思路

这道题考察的是简单分支问题，考察对if...else if...else的使用，当然也可以使用两层if...else来解决，当然对于一个分段函数，可能if...else if...else能更加清晰地表现出分段函数的情况，并且不容易出现问题。

AC代码

代码一：

```
#include <stdio.h>
int main()
{
    int x;
    scanf("%d", &x);
    if(x <= 0) printf("0");
    else if(x >= 6) printf("6");
    else printf("%d", x);
}
```

代码二：

```
#include <stdio.h>
int main()
{
    int x;
    scanf("%d", &x);
    if(x <= 0) printf("0");
    else
    {
        if(x >= 6) printf("6");
        else printf("%d", x);
    }
}
```

I. 算术平均分

解题思路

考察知识点：简单循环、运算。

思路：可以想到我们需要两个变量，一个储存总成绩，另一个储存有多少个成绩，然后每次读入一个成绩后更新这两个值，读入-1时跳出。最后输出总成绩/成绩个数（向下取整）即可。

C语言中整数除法默认是丢弃小数部分，例如5/2=2，(-7)/2=-3，所以对于正整数的除法默认就是向下取整。

AC代码

```
#include <stdio.h>

int main()
{
    int sum=0,cnt=0,m;
    while(1)
    {
        scanf("%d",&m);
        if(m==-1) break;
        cnt++;
        sum+=m;
    }
    printf("%d",sum/cnt);

    return 0;
}
```

J. 简单进制转换

解题思路

本题考察了循环语句。

本题需要使用数据读入的一种经典方式，即n个数据读入。我们可以利用while(n--)来方便的进行循环，当然for循环也可，对应格式为 for(i=0; i<n; i++);

对于进制转换，我们可以从高位逐个地进行遍历，首先乘以进制数，再加上当前这一位的数。这刚好和读入顺序相同，因此只需要边读入边计算即可。

AC代码

```
#include <stdio.h>

int main()
{
    int n, x, num = 0;

    scanf("%d", &n);
    while (n--)
    {
        scanf("%d", &x);
        num *= 2;
        num += x;
    }

    printf("%d\n", num);
```

```
    return 0;
}
```

K. 求最大公约数

解题思路与AC代码

求两数的最大公约数有多个不同的做法，注意到这道题hint里提到了由于时间限制可以暴力通过，这里介绍三种求最大公约数的办法，暴力法、辗转相除法、更相减损法，在学习了函数之后，同学们还可以使用Stein法求解。

这道题要求连续输入，常见的可用来连续输入的代码有以下几种：

1. `while((scanf(…)) != EOF)` …

2. `while(~(scanf(…)))… //~是取反的意思，对于1和0，~0 = 1，  ~1 = 0`

3.

```
while(1){
    …
    If(scanf(…) == EOF/* && …*/) break;
}
```

1、2两种代码比较相似，3可能应用于某些比较复杂的情况中。

同时连续输入要记得每一个结果后要换行。

暴力法：

思路：

我们从1开始进行枚举，一直枚举到输入的两数中较小的那个，看其中哪些数是输入的两数的共同因数，其中最大的就是要求的最大公因数。

代码：

```
#include<stdio.h>
int main()
{
    int a, b, max, i;
    while(scanf("%d %d", &a, &b) != EOF)
    {
        max = 0;
        i = 1;
        while(1)
        {
            if(a % i == 0 && b % i == 0)
            {
                if(i > max)
                {
                    max = i;
```

```
                }
            }
            i++;
            if(i > a || i > b)
            {
                break;
            }
        }
        printf("%d\n", max);
    }
    return 0;
}
```

辗转相除法

根据定理：两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数。我们可以通过反复进行大数对小数求余，直到小数正好就是大数的因数为止，此时我们所找到的小数就是我们要求的最大公因数。

```
#include <stdio.h>
int main()
{
    int a, b, temp;
    while((scanf("%d%d", &a, &b)) != EOF)
    {
        while(a != 0)
        {
            temp = a;
            a = b;
            b = temp;
            a %= b;
        }
        printf("%d\n", b);
    }
}
```

更相减损术

这是一个来自《九章算术》的算法"可半者半之，不可半者，副置分母、子之数，以少减多，更相减损，求其等也。以等数约之。" 第一步：若输入是两个偶数，则同除二，直到至少有一个不为偶数。 第二步：用大数减小数，直到差值和小数相等。 则第一步中约掉的若干个2的积与第二步中等数的乘积就是所求的最大公约数。

```
#include <stdio.h>
int main()
{
    int a, b, temp, mark = 0;
    int i;
    while((scanf("%d%d", &a, &b)) != EOF)
    {
        while(a % 2 == 0 && b % 2 == 0)
        {
            a /= 2;
            b /= 2;
            mark++;
```

```
        }
        while(a != b)
        {
            temp = a;
            a = a > b ? a : b;
            b = temp < b ? temp : b;
            a -= b;
        }
        for(i = 0; i < mark; i++) a *= 2;
        mark = 0;
        printf("%d\n", a);
    }
}
```

L. 诶乘闭

解题思路与AC代码

考查知识点：范围分析、运算技巧。（注：本题需要一定基础，可以选做）

思路：首先我们看到，尽管最终答案保证不超过30，但是中间答案可能会非常非常大（10^n级别），所以直接使用double 去储存答案肯定会超过限制。

因为最终答案范围的限制，我们可以推断出，给出的数据中，一定是一些大于1，一些小于1，所以一个想法就是在读入数据时分别存储大于1和小于1的数据，然后计算答案时，如果当前答案大于30（这个可以自己定一个标准），那么就乘一个小于1的数，反之乘一个大于1 的数，这样就可以保证中间答案在double范围之内。

```
#include <stdio.h>
#define maxn 100005

double f[maxn],g[maxn];
int ff,gg,i,j;

int main()
{
    double ans=1,x;
    int n;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%lf",&x);
        if(x>1) f[ff++]=x;
        else g[gg++]=x;
    }
    while(i<ff && j<gg)
    {
        if(ans>30) ans*=g[j++];
        else ans*=f[i++];
    }
    while(i<ff) ans*=f[i++];
    while(j<gg) ans*=g[j++];
    printf("%.2f",ans);
```

```
    return 0;
}
```

另一种比较有技巧性的思路就是，考虑用对数函数的性质把乘法转换为加法，这样也可以保证中间数据不超出double范围。代码如下：

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

int main()
{
    double ans=0,x;
    int n;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%lf",&x);
        ans+=log(x);
    }
    printf("%.2f",exp(ans));

    return 0;
}
```