A - Escape Character

难度	考点
1	转义字符

题目分析

本题主要让同学们了解 C 程序设计语言中的一些转义字符。根据题目所给提示，各转义字符的输出方法如下：

字符	输出方式
`"`	`\"`
`'`	`\'`
`\`	`\\`
`%`	`%%`

示例代码

```c
#include <stdio.h>

int main()
{
    printf("To output a quotation mark \", you should use \\\".\n");
    printf("To output a single quotation mark \', you should use \\\'.\n");
    printf("To output a slash /, you should use /.\n");
    printf("To output a backslash \\, you should use \\\\.\n");
    printf("To output a percent sign %%, you should use %%%%.\n");

    return 0;
}
```

B - Polynomial Interpolation

难度	考点
1	简单数学

题目分析

这个题目的背景是多项式插值。有兴趣的同学可以自行拓展相关内容。

由相关线性代数知识，有：

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 9^3 & 9^2 & 9 & 1 \\ 5^3 & 5^2 & 5 & 1 \\ 2^3 & 2^2 & 2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 12 \\ 1524 \\ 288 \\ 33 \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 9^3 & 9^2 & 9 & 1 \\ 5^3 & 5^2 & 5 & 1 \\ 2^3 & 2^2 & 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 12 \\ 1524 \\ 288 \\ 33 \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} -\frac{1}{32} & \frac{1}{224} & -\frac{1}{48} & \frac{1}{21} \\ \frac{1}{2} & -\frac{1}{28} & \frac{1}{4} & -\frac{5}{7} \\ -\frac{73}{32} & \frac{17}{224} & -\frac{29}{48} & \frac{59}{21} \\ \frac{45}{16} & -\frac{5}{112} & \frac{3}{8} & -\frac{15}{7} \end{bmatrix} \begin{bmatrix} 12 \\ 1524 \\ 288 \\ 33 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 7 \\ 3 \end{bmatrix}$$

即表达式为 $f(x) = 2x^3 + 0x^2 + 7x + 3$。

示例代码

```c
#include <stdio.h>

int main()
{
    int x;
    scanf("%d", &x);
    printf("%d", x * (2 * x * x + 7) + 3);

    return 0;
}
```

C - 一起集赞吧

难度	考点
2	简单 `if...else` 判断，输入输出

题目分析

题意即为输入两个整数 n, m，采用 `if-else` 分支结构判断它们的大小并输出不同的提示信息。

注意"集赞集够"是指 $n \geq m$，不是 $n > m$。

示例代码

```c
#include <stdio.h>

int main()
{
    int n, m;
    scanf("%d%d", &n, &m);
    if (n >= m)
        printf("Wow! Delicious^_^");
    else
        printf("%d", m - n);

    return 0;
}
```

D - 大家一起刷 TD

难度	考点
2	简单循环

题目分析

第一学期应当输出 48 个 `"TD"`，第二学期 96 个 `"TD"`，以此类推，第 n 个学期，应当输出 $48n$ 个 `"TD"`。

为了让输出的相邻两个 `"TD"` 间有一个空格，可以输出相应数量的 `"TD "`（注意末尾有一个空格）。

示例程序

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);

    int i;
    for (i = 1; i <= 48 * n; i = i + 1)
    {
        printf("TD ");
    }

    return 0;
}
```

E - 素数判定（简单版）

难度	考点
2	简单循环

题目分析

从 2 开始枚举数字，判断能否整除 n。如果 $[2, n)$ 的数都不能整除 n，则 n 为素数，否则不是。

示例程序

```c
#include <stdio.h>

int main()
{
    int n, i, canDivide = 0;
    scanf("%d", &n);
    for (i = 2; i < n; i = i + 1)
        if (n % i == 0)
            canDivide = 1;
    if (canDivide)
        printf("No\n");
    else
        printf("Yes\n");

    return 0;
}
```

F - /ˈæski/

难度	考点
3	分支结构嵌套、字符编码

题目分析

题面中的信息已经相当完善，此处不做分析。本题按照题给提示用若干个 `if` 判断相应情况并输出即可。

示例代码

示例代码一（利用题中给出的标准码表）

- 这里 `puts(_Str);` 用于输出一个字符串 `_Str` 并自动追加换行。
- `if` 里条件比较复杂时可分行对齐写。

```c
#include <stdio.h>

int main() {
    int c;
    scanf("%x", &c);
    if ((0x00 <= c && c <= 0x1F) || c == 0x7F)
    {
        puts("Control Character!");
        if (c == 0x00)
            puts("Null");
        else if (c == 0x09)
            puts("Horizontal Tab");
        else if (c == 0x0A)
            puts("Line Feed");
        else if (c == 0x0D)
            puts("Carriage Return");
        else
            puts("Other Control Character");
    }
    else if
    (
        (0x20 <= c && c <= 0x2F)
     || (0x3A <= c && c <= 0x40)
     || (0x5B <= c && c <= 0x60)
     || (0x7B <= c && c <= 0x7E)
    )
    {
        puts("Symbol Character!");
        if
        (
            c == '(' || c == ')'
         || c == '[' || c == ']'
         || c == '{' || c == '}'
         || c == '<' || c == '>'
        )
```

```
        {
            puts("Bracket");
        }
        else
            puts("Other Symbol");
    }
    else if (0x30 <= c && c <= 0x39)
    {
        puts("Digit Character!");
        putchar(c);
    }
    else if (0x41 <= c && c <= 0x5A)
    {
        puts("Upper Case Alphabet!");
        putchar(c);
    }
    else if (0x61 <= c && c <= 0x7A)
    {
        puts("Lower Case Alphabet!");
        putchar(c);
    }

    return 0;
}
```

示例代码二（使用 `ctype.h` 头文件）

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

int main()
{
    int c;
    scanf("%x", &c);
    if (iscntrl(c))
    {
        puts("Control Character!");
        if (c == '\0')
            puts("Null");
        else if (c == '\t')
            puts("Horizontal Tab");
        else if (c == '\n')
            puts("Line Feed");
        else if (c == '\r')
            puts("Carriage Return");
        else
            puts("Other Control Character");
    }
    else if (isdigit(c))
    {
        puts("Digit Character!");
        putchar(c);
    }
    else if (isupper(c))
    {
        puts("Upper Case Alphabet!");
```

```
        putchar(c);
    }
    else if (islower(c))
    {
        puts("Lower Case Alphabet!");
        putchar(c);
    }
    else
    {
        puts("Symbol Character!");
        if (c == '(' || c == ')' || c == '[' || c == ']' || c == '{' || c == '}' || c == '<' || c == '>')
            puts("Bracket");
        else
            puts("Other Symbol");
    }

    return 0;
}
```

G - Crafting Table

难度	考点
4	数学思维、多组数据的读入

题目分析

首先，本题的数据范围为 $1 \leq t \leq 10^5$， $0 \leq n, m \leq 10^9$，直接暴力模拟的计算量在 10^{14} 数量级，评测机每秒的计算量在 10^8 到 10^9 数量级，因此这样做会超时，得到 `TLE` 的评测结果，我们需要找到一个可以快速计算的数学表达式以减少计算量。

其次，我们可以发现只考虑钻石铲和钻石剑是最优的，因为其它三种工具都会比上述两种工具之一耗费更多的原材料。

最后，每制作 1 个工具至少耗费 1 个木棍和 1 个钻石，每制作 1 个工具耗费钻石和木棍的总量为 3，因此最大值不超过 $\min(n, m, \lfloor \frac{n+m}{3} \rfloor)$。（其中 $\lfloor x \rfloor$ 表示 x 的向下取整）

下面我们证明最大值即为 $\min(n, m, \lfloor \frac{n+m}{3} \rfloor)$：

- 若 $2n \leq m$，因为每制作 1 个工具至少耗费 1 个木棍，所以此时答案为 n，且 $n \leq \lfloor \frac{n+m}{3} \rfloor \leq m$。
- 若 $2m \leq n$，因为每制作 1 个工具至少耗费 1 个钻石，所以此时答案为 m，且 $m \leq \lfloor \frac{n+m}{3} \rfloor \leq n$。
- 若 $2n > m$ 且 $2m > n$，不断重复下述过程直至 $n + m < 3$：若 $n \leq m$，则制作一把钻石剑，否则制作一把钻石铲。$2n > m$ 且 $2m > n$ 的条件保证上述过程中 n 和 m 始终不为 0，所以此时答案为 $\lfloor \frac{n+m}{3} \rfloor$，且 $\lfloor \frac{n+m}{3} \rfloor \leq \min(n, m)$。

示例程序

```c
#include <stdio.h>

int main()
{
    int t, n, m, ans, i = 1;
    scanf("%d", &t);
    while (i <= t)
    {
        scanf("%d%d", &n, &m);
        ans = (n + m) / 3;
        if (n < ans) ans = n;
        if (m < ans) ans = m;
        printf("%d\n", ans);
        i = i + 1;
    }

    return 0;
}
```

H - 会砍价的小翔哥

难度	考点
2	浮点数计算，简单循环，格式化输出

题目分析

循环输入雪糕价格 x，利用强制类型转换 `(int)x` 即可得到价格的整数部分，那么 `x - (int)x` 就是价格的小数部分，累加即可。

示例程序

```c
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);

    float x;
    float sum = 0.0; // float类型变量sum用来累计省下来的钱，初始化为0。
    int i;
    for (i = 0; i < n; i = i + 1)
    {
        scanf("%f", &x);
        sum = sum + (x - (int)x);
    }

    printf("%.2f", sum); // 格式化输出，%.2f表示保留两位小数

    return 0;
}
```

I - 挨乘闭加吸

难度	考点
4	数学思维、简单循环

题目分析

此题本意并非枚举 A, B, C 判断，数据范围为 10^6 ，暴力复杂度 $O(N^2)$ 到 $O(N^3)$ 不等，一定会得到 `TLE` 的结果

正解

每一个小于 N 的 $A \times B$ 都有唯一确定的 C 与之对应，所以只需考虑 (A, B) 的数量即可。

枚举 1 到 $N-1$ 作为 A，如果 N 不能整除 A，那么相应 B 的个数为 $\lfloor N/A \rfloor$，否则为 $\lfloor N/A \rfloor - 1$。

示例程序

```c
#include <stdio.h>

int main()
{
    int n, ans;
    int i;
    scanf("%d", &n);
    for (i = 1; i < n; i = i + 1)
    {
        ans += n / i;
        if (n % i == 0) --ans;
        // 或上面两句写成 ans += n % i == 0 ? n / i - 1 : n / i;
    }
    printf("%d", ans);

    return 0;
}
```

J - RSA 的钥匙（简单版）

难度	考点
3	循环结构，判断结构

题目分析

本题需要根据题目给出的步骤进行模拟，并用二重循环结构完成任务。对于判断 $\gcd(e, \varphi) = 1$，可以采用第一节课的课件提到的方法求出 e 和 φ 的 GCD。

示例程序

```c
#include <stdio.h>

int main()
{
    int p, q, n, phi;
    int e, d, gcd;

    scanf("%d%d", &p, &q);
    n = p * q;
    phi = (p - 1) * (q - 1);

    for (e = 2; e < phi; e = e + 1)
    {
        gcd = e;
        while (!(e % gcd == 0 && phi % gcd == 0))
            gcd = gcd - 1;
        if (gcd == 1)
        {
            for (d = 2; d < phi; d = d + 1)
                if (e * d % phi == 1)
                    printf("(%d,%d),(%d,%d)\n", n, e, n, d);
        }
    }

    return 0;
}
```

K - 素数判定

难度	考点
3	简单循环，数据类型

题目分析

从 2 开始枚举数字，判断能否整除 n。如果 $[2, \lfloor \sqrt{n} \rfloor]$ 的数都不能整除 n，则 n 为素数，否则不是。

注意，int 范围无法表示 10^{14} 大小的数，使用 `long long` 读入和存储。

示例程序

```c
#include <stdio.h>

int main()
{
    long long n, i;
    int canDivide = 0;
    scanf("%lld", &n);
    for (i = 2; i * i <= n; i++)
        if (n % i == 0)
            canDivide = 1;
    if (canDivide)
        printf("No\n");
    else
        printf("Yes\n");

    return 0;
}
```

L - 奶茶时间

难度	考点
3	输出格式控制、`if-else` 条件判断

题目分析

按题目完成减法计算和时间早晚的比较即可。常见错误：

1. 减法的进位出错；
2. 漏写了某个触发“秃头警告”的判断条件。

示例程序

```c
#include <stdio.h>

int main()
{
    int a, b, c, d, e, f, g, h;
    scanf("%d:%d", &a, &b);
    scanf("%d:%d", &c, &d);
    scanf("%d:%d", &e, &f);
    h = d - f - 1;
    if (h < 0)
    {
        h += 60;
        g = c - e - 1;
    }
    else
        g = c - e;
    if (g < a || (g == a && h < b))
        printf("Hair Loss Warning");
    else
        printf("%02d:%02d", g, h);

    return 0;
}
```

M - 税（普通版）——增值税

难度	考点
3	浮点数计算、多组数据输入、输出格式控制

题目分析

只需按照题目要求计算每个商品的相关数据即可，注意数据类型即可。

示例程序

```c
#include <stdio.h>

int main()
{
    char name[50];
    int Class, Num, TaxLevel;
    double Money, Cost, AllMoney, Tax;
    while (~scanf("%s%d%d%lf", name, &Class, &Num, &Money))
    {
        switch (Class)
        {
        case 1:
        case 3:
            TaxLevel = 9;
            break;
        case 2:
            TaxLevel = 13;
            break;
        case 4:
            TaxLevel = 6;
            break;
        }
        AllMoney = Money * 100 / (100 + TaxLevel);
        Cost = AllMoney / Num;
        Tax = Money - AllMoney;
        printf("%s %d %.2lf %.2lf %d%% %.2lf\n", name, Num, Cost, AllMoney,
TaxLevel, Tax);
    }

    return 0;
}
```

N - 书号的秘密

难度	考点
3	条件判断、字符串处理

题目分析

只需按要求处理每一位数并做最后判断即可，两种处理方式见示例程序。

示例程序1：作为字符串处理

```c
#include <stdio.h>

int main()
{
    char s[233];
    int res, i, j;
    while (scanf("%s", s + 1) != EOF)
    {
        res = 0;
        j = 0;
        for (int i = 1; i <= 11; i++)
        {
            if (s[i] == '-') continue;
            res += (s[i] - '0') * j;
            ++j;
        }
        if (res % 11 == s[13] - '0')
            puts("YES");
        else
            puts("NO");
        // 或上面写成 puts(res % 11 == s[13] - '0' ? "YES" : "NO");
    }

    return 0;
}
```

示例程序2：作为多位数处理

```c
#include <stdio.h>

int main()
{
    int a, b, c, d;
    int sum;
    while (scanf("%d-%d-%d-%d", &a, &b, &c, &d) != EOF)
    {
        sum = a;
        sum += 2 * (b / 1000);
        sum += 3 * ((b / 100) % 10);
        sum += 4 * ((b / 10) % 10);
```

```
        sum += 5 * (b % 10);
        sum += 6 * (c / 1000);
        sum += 7 * ((c / 100) % 10);
        sum += 8 * ((c / 10) % 10);
        sum += 9 * (c % 10);
        if (sum % 11 == d)
            puts("YES");
        else
            puts("NO");
        // 或上面写成 puts(sum % 11 == d ? "YES" : "NO");
    }

    return 0;
}
```

O - 归零

难度	考点
3	贪心

题目分析

本题是一道简单的贪心题目，分为两大种情况考虑：

- 两数异号，注意到第二种运算并无必要，可以直接用他们的绝对值相加乘 x 即可。
- 两数同号，首先将他们均变为绝对值。很多同学在判断负数的情况时出错，这里建议大家直接变为相反数处理），再进行讨论。
  - 如果 $2 \cdot x < y$，那么直接使用第一种操作最省电，直接用他们的绝对值相加乘 x 即可。
  - 如果 $2 \cdot x \geq y$，那么先使用第二种操作将最小数归零，然后用第一种操作将最大数剩余部分归零。

另外需要注意本题的答案范围超过了 `int`，请使用 `long long` 保存变量和答案。

示例程序

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

int main()
{
    long long a, b, x, y;
    scanf("%lld%lld%lld%lld", &a, &b, &x, &y);
    if (a < 0 && b < 0)
    {
        a = -a;
        b = -b;
    }
    if (a * b < 0 || x * 2 <= y)
    {
        printf("%lld", x * (llabs(a) + llabs(b)));
        return 0;
    }
    else if (a < b)
        printf("%lld", a * y + (b - a) * x);
    else
        printf("%lld", b * y + (a - b) * x);

    return 0;
}
```

P - 排好序了吗

难度	考点
4	排序，运行内存概念

题目分析

本题实际上是串行输入给定一个数组，你需要判断这个数组是否是单调不降或是单调不升的。但是题目的空间不允许我们存下整个数组后再判断，因此需要改变策略，仅通过每次比较相邻两项满足何种序列情况，进而进行维护。

具体来说，我们可以用两个变量 `inc` 和 `dec` 分别表示当前的序列是否满足单调不降和单调不升，同时记录当前序列读入的最后一个值 `pre`，则每次读入一个新的数 `now` 时：

1. 若 `pre > now`，则当前序列不可能单调不降，令 `inc = 0`；
2. 若 `pre < now`，则当前序列不可能单调不升，令 `dec = 0`；
3. 更新当前序列最后一个值为 `now`，即 `pre = now`。

最后，只要 `inc` 和 `dec` 中至少有一个满足，则序列就是有序的；否则序列无序。

示例程序

```c
#include <stdio.h>

int main()
{
    int inc = 1, dec = 1;
    int pre, now;
    scanf("%d", &pre);
    while (scanf("%d", &now) != EOF)
    {
        if (pre > now) inc = 0;     // 不满足单调不降
        if (pre < now) dec = 0;     // 不满足单调不升
        pre = now;                  // 更新上一个值
    }
    if (inc == 1 || dec == 1)
        printf("Beautiful");
    else
        printf("No");

    return 0;
}
```

Q - Easy PrimeSmash

难度	考点
4	循环求和、条件判断、变量初始化

题目分析

按照题目描述所给的提示进行计算和判断即可。常见错误如下：

1. 有多组数据输入，忘记了必要的变量初始化，导致单组数据正确，多组数据出错的情况。
2. 在 `if` 有结果后立即结束 `for` 循环，而没有完全读入当前组的数据，导致不能正确地读入后续的数据。
3. 没有依照题目要求（`ans` 尽量小）判断答案，`if` 语句的顺序不正确。（示例程序中的做法是：用并列的 `if` 语句，并先判断较大的可能答案，后判断较小的可能答案，让之后的赋值覆盖掉前面的值。）
4. 有的同学在“从个位起每三位截成一段”的计算中，忽略了最高位位数不足三位的数字的计算”。（有的同学分成了六位数字一组，错误原因同理） 另一种做法是忽略题目描述所给的提示，利用模运算的特点，在求和的过程中不断取模，最后判断每个模数是否是待判断的数字的约数。

示例程序

```c
#include <stdio.h>

int main()
{
    int n, x, sum, firstnum, ge, shi, bai, flag, total, pos, i, ans;
    while (scanf("%d", &n) == 1)
    {
        sum = 0;
        ge = 0;
        shi = 0;
        bai = 0;
        pos = 0;
        total = 0;
        flag = 1;
        for (i = 0; i < n; i++)
        {
            scanf("%d", &x);
            if (i == 0) firstnum = x;
            sum += x;
            switch (pos % 3)
            {
            case 0:
                ge = x;
                break;
            case 1:
                shi = x;
                break;
            case 2:
                bai = x;
                total += flag * (bai * 100 + shi * 10 + ge);
                ge = 0;
```

```
                shi = 0;
                bai = 0;
                flag *= -1;
                break;
            }
            pos++;
        }
        total += flag * (bai * 100 + shi * 10 + ge);
        ans = 0;
        if (total % 13 == 0) ans = 13;
        if (total % 11 == 0) ans = 11;
        if (total % 7 == 0) ans = 7;
        if (firstnum % 5 == 0) ans = 5;
        if (sum % 3 == 0) ans = 3;
        if (firstnum % 2 == 0) ans = 2;
        if (ans == 0)
            printf("qwq\n");
        else
            printf("Smash it with %d!\n", ans);
    }

    return 0;
}
```

R - 小狗过河

难度	考点
5	贪心

题目分析

本题是一道比较直观的贪心题。

可以发现，我们应该先考虑最重的狗比较方便（因为体重大的狗要么一只狗孤独地走，要么带一只轻狗）。一个最优方案是：如果当前最重的狗 y 能和当前最轻的狗 x 一起过河，那么就安排他们一起，否则只让 y 单独过河。下面证明这种方案得到的一定是最优解。

假设 $a[x]+a[y]>w-m$，那么无论哪种方案，y 都只能单独过河。

假设 $a[x]+a[y]\le w-m$，即 y 可以带一条狗一起过河，那么 y 必须带上一只（因为带一定比不带要好）。如果有很多只轻狗都可以和 y 一起过河，下面证明带最轻的 x 可以得到最优解。

设某种最优方案下，过河组队为 $(x,y'),(x',y)$ $(a[x]\le a[x']\le a[y']\le a[y])$，即

$$a[x']+a[y]\le w-m \qquad (1)$$

$$a[x]+a[y']\le w-m \qquad (2)$$

由于 $a[x]+a[y]\le w-m$ (假设的条件) 可知 (x,y) 可以过河。而由公式 (1) 有

$$a[x']+a[y']\le a[x']+a[y]\le w-m$$

即 (x',y') 的方案也可以过河。也就是说交换之后，我们选择最轻的 x 过河的答案和最优解的答案相同，所以我们可以直接选择最轻的狗 x。

每次选择后剩余的狗的数量为 $n-1$ 或 $n-2$，重复这个过程直到送走所有的狗狗。

示例程序

```cpp
#include <stdio.h>

int a[100000 + 5];  // 由于数组长度较大，故放在 main 函数外作为全局数组

int main()
{
    int n, m, w;
    scanf("%d%d%d", &n, &m, &w);
    w -= m;

    int i, j, ans = 0;
    for (i = 1; i <= n; ++i) scanf("%d", &a[i]);
    i = 1;
    j = n;
    while (i <= j)
    {
        if (a[i] + a[j] <= w) ++i;
        --j;
        ++ans;
    }
```

```
    printf("%d", ans);

    return 0;
}
```

S - 和求列数

难度	考点
5	逆向思维、循环与条件判断

题目分析

正向考虑非常困难，因为`求和操作`与`翻转操作`的顺序和数量都是未知的，因此我们需要逆向考虑来解决这道题。

对于题目中给出的两种操作，我们可以观察出以下四点：

- 对于一个**正整数**数组，无论做多少次`求和操作`或是`翻转操作`，它依旧是一个**正整数**数组。
- 对于一个**正整数**数组，`求和操作`后它会变为一个**严格递增**的**正整数**数组，此时对它再进行一次`翻转操作`，它将变为一个**严格递减**的**正整数**数组。
- `求和操作`的逆向操作为：将 $[a_1, a_2, \cdots, a_n]$ 变为 $[a_1, (a_2 - a_1), \cdots, (a_n - a_{n-1})]$。对于一个**正整数**数组，逆向`求和操作`后它仍是一个**严格递增**的**正整数**数组当且仅当它是**严格递增**的。
- `翻转操作`的逆向操作还是`翻转操作`，偶数次`翻转操作`等价于不操作。

从以上四点，我们可以通过下面的方法快速求得对于数次操作后的数组 b，它通过逆向的`求和操作`与`翻转操作`所能得到的所有数列：

1. 我们可以对 b 进行一次逆向的`翻转操作`得到一个新的数列。
2. 如果 b 是**严格递增**的，那么我们可以对其进行一次逆向的`求和操作`，得到一个新的数列，并返回步骤1。
3. 如果 b 是**严格递减**的，那么我们可以对其进行一次逆向的`翻转操作`后再进行一次逆向的`求和操作`，得到一个新的数列，并返回步骤1。
4. 如果 b 既不**严格递增**也不**严格递减**，此后 b 无法通过任何操作得到一个之前未出现过的数列，因此可以终止程序。

我们只需模拟上述过程，记录操作次数，并且每得到一个新的数列就判断是否和 a 相同，如果相同则输出此时的操作次数并终止程序；如果直到最后也没有找到和 a 相同的数列，应输出 -1。

最后，`求和操作`的增长速度是 x^{n-1} 级别的，其中 n 是数列长度，因此在本题的数据范围下，上述算法不会超时，可以参考下面的表格：

数列长度 n	求和操作次数的最大值
3	1414212
4	18169
5	2211
10	85
100	9
1000	5
100000	3

示例程序

```c
#include <stdio.h>

long long a[100086], b[100086];

int main()
{
    int t, n;
    int tag, ans;
    int x, y;
    int i, j = 1;
    int tmp;

    scanf("%d", &t);
    while (j <= t)
    {
        scanf("%d", &n);
        for (i = 1; i <= n; i++) scanf("%lld", &a[i]);
        for (i = 1; i <= n; i++) scanf("%lld", &b[i]);
        tag = 0, ans = 0;
        while (1)
        {
            for (i = 1; i <= n; i++)
            {
                if (a[i] != b[i]) break;
                if (i == n) tag = 1;
            }
            if (tag == 1) break;
            for (i = 1; i <= n; i++)
            {
                if (a[n - i + 1] != b[i]) break;
                if (i == n) tag = 1, ans++;
            }
            if (tag == 1) break;
            x = 0, y = 0;
            for (i = 2; i <= n; i++)
            {
                if (b[i] >= b[i - 1]) x = 1;
                if (b[i] <= b[i - 1]) y = 1;
            }
            if (x == 1 && y == 1) break;
            if (y == 1)
            {
                ans++;
                for (i = 1; i <= n - i + 1; i++)
                {
                    tmp = b[i], b[i] = b[n - i + 1], b[n - i + 1] = tmp;
                }
            }
            ans++;
            for (i = n; i; i--) b[i] -= b[i - 1];
        }
        printf("%d\n", tag == 1 ? ans : -1);
        j++;
    }
```

```
55
56      return 0;
57  }
```