C1-solution

A 向着星辰与深渊 ！

题意分析

欢迎同学们来到联盟课的第一次上机赛！ 输出 `Ad astra abyssosque !` 即可。复制 `Hint` 的代码就可以通过，同时也提醒大家看题不要忽视 `Hint` 的部分。

示例代码

```
#include <stdio.h>
int main()
{
    printf("Ad astra abyssosque !");
    return 0;
}
```

B 小猫咪又知道什么呢

题意分析

输出小猫咪的字符画，只需注意转义字符与换行符的使用。 输出 `\` 时需要进行转义，使用 `\\` 来输出。

示例代码

多行 `printf`

```
#include <stdio.h>
int main()
{
    printf(" __^__^____\n");
    printf(" /_(\"`o `)_/\\\n");
    printf("/| U U |/\\\n");
    printf(" |_________|/\n");
    return 0;
}
```

单行 `printf` ，使用 `\` 来使下一行承接上一行

```
#include <stdio.h>
int main()
{
    printf(" __^__^____\n\
/_(\"`o `)_/\\\n\
/| U U |/\\\n\
|_________|/\n");
    return 0;
}
```

C Humagear的梦想!

题意分析

只需要判断 s 和 a 的大小即可。要注意输出的具体内容，推荐直接复制。

示例代码

```
#include <stdio.h>
int main()
{
    int s, a;
    scanf("%d%d", &s, &a);
    if (s < a)
        printf("W1N\n");
    else
        printf("L0SE\n");
    return 0;
}
```

D 字符的距离

题意分析

我们知道每个字符都对应一个取值 0~127 的 ASCII 编码值。要求编码值之差，直接将两个字符类型的变量相减即可。char 类型的变量也可以像 int 类型一样进行加减乘除取余等运算，但是要注意 char 只能表示 -128~127 的整数。

题目并不保证 $C_1 < C_2$ ，因此勿忘判断大小关系。

需要注意的是许多同学会使用 `scanf("%c %c",&c1,&c2);` 这种方式读入用空格分开的两个字符，但当我们把空格本身作为第二个需要读入的字符时就会引发问题。这是因为 scanf 函数中的 "%c %c" 内的一个空格并不是表示一个空格把两个输入字符分开，而是表示忽略下一个非空白字符（Whitespace character）前的所有字符！因此无论我们想让第二个 %c 读入空格还是回车还是其他制表符，都会被 scanf 吞掉。我们可以通过 `scanf("%c%c%c",&c1,&c2,&c2);` 或者直接用 getchar 读入来规避这个问题。

关于 scanf 的更多特性，参见https://cplusplus.com/reference/cstdio/scanf/。同学们在之后的字符、字符串读入过程中可能遇到更多类似问题。

示例代码

```
#include<stdio.h>
int main()
{
    int c1,c2;
    c1=getchar(); //getchar()返回char 直接在赋值过程中转换成int
    c2=getchar(); //当然也可以直接两个char相减
    c2=getchar();
    if(c1-c2>0) printf("%d\n",c1-c2);
    else printf("%d\n",c2-c1);
    return 0;
}
```

E cbd求和!

难度	考点
1	循环，输入输出

题目分析

考察简单的for循环和取模运算。

根据题目，我们需要读入所有的数字，并且判断奇偶性，是奇数，则累加。判断奇偶性的方法很多，如下：

- 取模：a%2≠0，说明a不为偶数。
- 位运算：(a&1)≠0，则说明a的二进制最低位为1，a为奇数。（可以了解，后面会学到）

参考代码

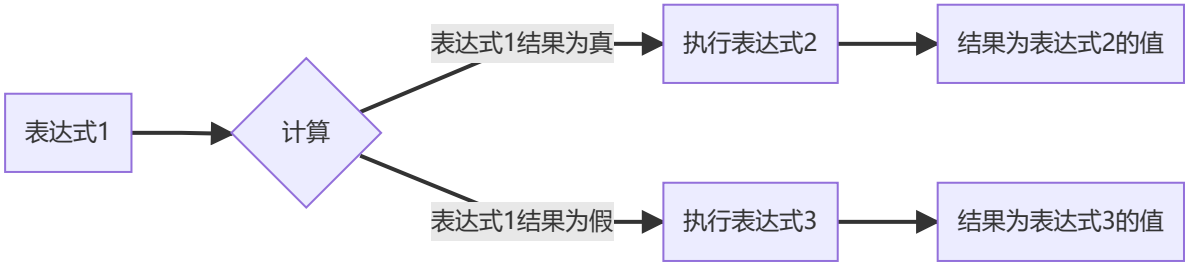
```
#include<stdio.h>

int main()
{
    int n,tmp,sum=0;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&tmp);
        if(tmp%2)//这里的条件可以修改成 tmp&1
            sum+=tmp;
    }
    printf("%d\n",sum);
    if(sum%2)
        printf("no\n");
    else
        printf("yes\n");

}
```

三目运算符

语法如下：

```
表达式1?表达式2:表达式3;
```

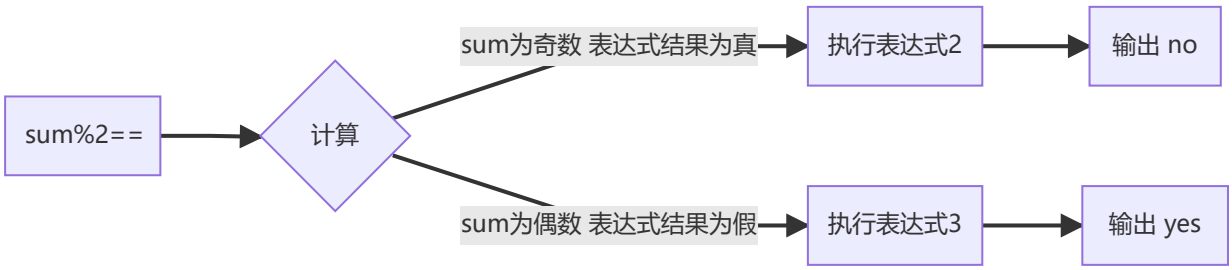
语义如下：



所以，本题的输出也可不采用条件判断，可直接用**三目运算符**输出，如下：

```
printf("%d\n%s\n",sum,sum%2==1?"no":"yes");
```

理解如下：



F 简单的位运算2023

题意分析

本题属于简单题，计算3个有符号的 `long long` 异或的结果.需要注意数据范围，输入输出 `long long` 用 `%lld`

参考代码

```
#include <stdio.h>
int main()
{
    long long a, b, c;
    scanf("%lld%lld%lld", &a, &b, &c);
    printf("%lld\n", a ^ b ^ c);
    return 0;
}
```

G 花车颠啊颠...

题意分析

本题要求倒序输出，只需用数组存储后下标从大到小输出即可，注意不定组数据读入的方法。

示例代码

```
#include <stdio.h>
int main()
{
    int a[1005], n = 0, b;
    while (scanf("%d", &b) != EOF) // 多组数据读入
        a[n++] = b;
    n--; // 注意这个 n 为最后一个读入的数的下标+1
    while (n--)
        printf("%d ", a[n]);
    return 0;
}
```

H 青春AC少年不会梦到数值分析!

考点	难度
浮点数比较	1

题意说明

按照题目意思，我们需要计算表达式的值并比较两者差异，关键点如下：

1. 计算表达式的值由于涉及到平方，注意使用 `long long int` 或 `double` 数据类型防止数据溢出。
2. 比较两个浮点数的差异一般是一个固定书写模式，比较两个浮点数是否相同时也可以使用这个板子，建议大家熟练或写宏定义

> ps：希望同学们能够观察到两种计算方式带来的结果的差异，一般地，第二种方式计算精度高，这是因为计算机计算绝对值极大而相差极小的两数相减时，会存在极大的舍入误差，因而可以通过改写表达式计算。斗虫大师认为数值分析是一项很有用的利器，所以他希望自己能认真学习，强化基础，也希望能够把分析转化的思维带给大家。

示例代码

```
#include<stdio.h>
#include<math.h>
#define absm(x, y) fabs((x)-(y)) //宏定义，可以根据个人需求进一步优化
int main() {
    double x; //long long int 也可以
    while(scanf("%lf",&x)≠EOF)
    {
        printf("%.6lf %.6lf\n", sqrt((x*x + 1))-x, 1.0/(sqrt((x*x + 1))+x));
        printf("%.12lf %.12lf\n", sqrt((x*x + 1))-x, 1.0/(sqrt((x*x + 1))+x));
        if(absm(sqrt((x*x + 1))-x, 1.0/(sqrt((x*x + 1))+x)) < 1e-12)
        printf("accepted\n");
    }
    return 0;
}

//比较两个浮点数a,b是否相同:
// if(fabs(a-b)<eps)
// ……
```

I ljh的 M 数列! （初学者不做要求）

难度	考点
4	循环、数组

问题分析

本题主要考察数组的应用，我们很容易能想到通过双重循环直接枚举子序列的左右端点，再判断选取的子序列是否为 **M 序列**，但观察数据范围后，我们发现这种方法的时间复杂度 $O(n^2)$ 甚至 $O(n^3)$ 并不符合题目要求（会超时）。

考虑如何优化，我们可以先固定左端点，然后枚举右端点的同时根据特定的条件对左端点进行右移更新。那么更新的依据是什么呢？考虑到 **M 序列**要求所有数都至少出现过 1 次，所以当左端点上的数出现**超过** 1 次时，我们可以认为该点冗余了，可以将左端点右移以缩短我们选取的子序列的长度，通过循环不断右移左端点，直到左端点上的数出现次数为 1 ，我们可以形象的将其理解为一个**滑动窗口**。

再枚举从左到右依次枚举右端点的过程中，我们可以通过一个数组 ap 统计枚举到的数的出现次数，再通过一个变量 q ，统计当前子序列中出现的数的种数，当 $q == m$ 时，意味着我们找到了一个 **M 序列**，此时就可以比较更新 len 值，最后输出最小的 len 值即为答案，这样我们就在 $O(n)$ 的时间复杂度内解决了这个问题。

参考代码

```
#include <stdio.h>
int n, m;
int a[500010], ap[500010] = {0}; // a 数组存储序列，  ap 数组统计出现次数
int main()
{
    scanf("%d%d", &n, &m);
    int q = 0, first = 1, len = 500001;
    // first 为初始左端点，将 len 的初始值设置为极大值
    for (int i = 1; i ⩽ n; i++) // i 代表枚举的右端点
    {
        scanf("%d", &a[i]); // 输入当前右端点的数 a[i]
        if (!ap[a[i]]) q++; // 如果 a[i] 之前没有出现过，令 q+1
        ap[a[i]]++; // 更新a[i]出现次数
        for (first; first ⩽ n; first++) // 更新左端点 first
        {
            if (ap[a[first]] ⩽ 1) break;
            // 如果左端点的数 a[first]出现次数 ⩽1 那么停止更新
            else ap[a[first]]--;
            // 否则左端点的数出现次数 -1 并令左端点右移 1 位（for循环条件）
        }
        if (q == m && i - first + 1 < len)
        // 如果出现的数的种数 ==m 并且当前子序列长度 i-first+1 比我们之前保存的M序列长度 len 小，
更新 len
            len = i - first + 1;
    }
    if (q ≠ m) printf("Not Find");
    // 如果枚举完整个数列后出现的数的种数仍然不等于 m 则表示给定序列没有M序列 输出 Not Find
    else printf("%d", len); // 找到了M序列，输出最短长度 len
    return 0;
}
```

J 来玩生命游戏吧 （初学者不做要求）

问题分析

本题只需要照着题目里面说的步骤一步一步模拟即可。

在计算“周围有多少个活细胞”时需要注意一下目前正在计算的细胞是否超过了边界等不需要计算的情况。

在代码中，使用了 $cnt[i,j]$ 记录了第 i 行第 j 列的细胞周围有多少个活细胞，然后直接更新棋盘 $m[i,j]$ 即可。

参考代码

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <ctype.h>
#include <stdlib.h>
#define ll long long
#define maxn 100 + 5
const double eps = 1e-11;
int m[maxn][maxn];
int cnt[maxn][maxn];
int n, k;
int cn(int i, int j)
{
    int ans = 0;
    for (int si = i - 1; si ⩽ i + 1; ++si)
    {
        for (int sj = j - 1; sj ⩽ j + 1; ++sj)
        {
            if (si ⩾ 1 && si ⩽ n && sj ⩾ 1 && sj ⩽ n && !(si == i && sj == j))
                ans += m[si][sj];
        }
    }
    return ans;
}
int main()
{
    scanf("%d%d", &n, &k);
    for (int i = 1; i ⩽ n; ++i)
        for (int j = 1; j ⩽ n; ++j)
            scanf("%d", &m[i][j]);
    while (k--)
    {
        memset(cnt, 0, sizeof(cnt));
        for (int i = 1; i ⩽ n; ++i)
        {
            for (int j = 1; j ⩽ n; ++j)
            {
                cnt[i][j] = cn(i, j);
            }
        }
        for (int i = 1; i ⩽ n; ++i)
        {
            for (int j = 1; j ⩽ n; ++j)
            {
                if (m[i][j] == 0)
                {
                    if (cnt[i][j] == 3)
```

```
                        m[i][j] = 1;
                }
                else
                {
                    if (cnt[i][j] < 2 || cnt[i][j] > 3)
                        m[i][j] = 0;
                }
            }
        }
    }
    for (int i = 1; i ⩽ n; ++i)
    {
        for (int j = 1; j ⩽ n; ++j)
            printf("%d ", m[i][j]);
        printf("\n");
    }
    return 0;
}
```