1.小朋友的年龄

> n个小朋友们按照年龄排好队（注意：老师只要求年龄相同的小朋友站在一起）。请统计一下小朋友们有多少种不同的年龄，并按照小朋友们排队的顺序输出这些年龄。

输入

> 输入两行：
> 第一行是n（1 <= n <= 1000）。
> 第二行是n个整数，邻近两数之间用一个空格隔开。

输出

> 输出两行：
> 第一行是不同年龄的个数m（1 <= m <= n）。
> 第二行是按照小朋友们排队顺序输出的m个年龄，邻近两数之间用一个空格隔开。

```c
#include <stdio.h>
int main() {
    int n, m = 0, i, j,ages[1000], unique[1000];
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &ages[i]);
    }
    for (i = 0; i < n; i++) {
        int current = ages[i],found = 0;
        for (j = 0; j < m; j++) {
            if (unique[j] == current) {
                found = 1;
                break;
```

```
            }
        }
        if (!found) {
            unique[m++] = current;
        }
    }
    printf("%d\n", m);
    for (i = 0; i < m; i++) {
        printf("%d", unique[i]);
        if (i != m - 1) printf(" ");
    }
}
```

2.红蜻蜓的披风

> 红蜻蜓穿了一件漂亮的披风，它在湖上竖直向天上飞，漂亮的披风倒映在水面上。披风从上到下绣着多个字符，请你把披风上的字符与湖水中的倒影连起来，组成新的字符串。
>
> ---
>
> 输入
>
> > 输入一个字符串（1 <= 字符串长度 <= 99），以回车结束，代表红蜻蜓的披风。
>
> ---
>
> 输出
>
> > 输出字符串与字符串倒影的连接字符串。

```
#include <stdio.h>
#include <string.h>
void reverse_string(char *s) {
    int i, j;
```

```
    int len = strlen(s);
    for (i = 0, j = len - 1; i < j; i++, j--) {
        char temp = s[i];
        s[i] = s[j];
        s[j] = temp;
    }
}

int main() {
    char a[100];
    scanf("%s", a);
    printf("%s", a);
    reverse_string(a);
    printf("%s", a);
}
```

3.一年中的第几天

> 给定一个具体的日期，请输出，这一天是当年的第几天?
>
> ---
>
> 输入
>
> > 输入格式如下：
> > year-month-day
> > 如：
> > 1999-9-9
> > 代表1999年9月9日。
>
> ---
>
> 输出
>
> > 输出一个整数，代表这一天为那一年的第多少天。

```c
#include <stdio.h>
int main() {
    int year, month, day,i,j;
    int day_in_month[12] = {31, 28, 31, 30, 31, 30, 31, 31,
30, 31, 30, 31};
    scanf("%d-%d-%d", &year, &month, &day);
    if((year % 4 == 0 && year % 100 != 0)||year % 400 == 0)
{
        day_in_month[1] = 29;
    }
    for (i = 0; i < month - 1; i++) {
        day += day_in_month[i];
    }
    printf("%d", day);
}
```

4.移动

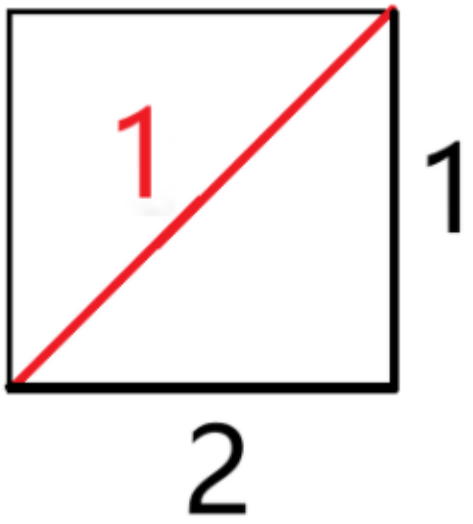
> 对弈，是国人引以自豪的事情，所以，各种棋的玩儿法更是层出不穷。
> 本题是一个极为简单的玩法。
> 两个人只玩一个棋子，棋子开始时在棋盘的右上角，棋盘的大小为m行n列。即棋子待在的位置是（1, n）。在这个棋盘上，棋子只能够向左、左下（斜线）和下方移动。
> 作为对弈的双方，每人每一步必须将棋子挪动，直到一方不能挪动的棋子为止。不能挪动的一方为输方。在告诉你m和n后，你可以完成这个任务吗？如果第一人胜利，则结果为first!，第二人胜利结果为second!
> 本题，假设两人都是极为聪明，不会走错（即均遵循最佳策略）。

输入

> 输入两个正整数m和n（0 < m，n <= 2000），两数间用一个空格隔开。

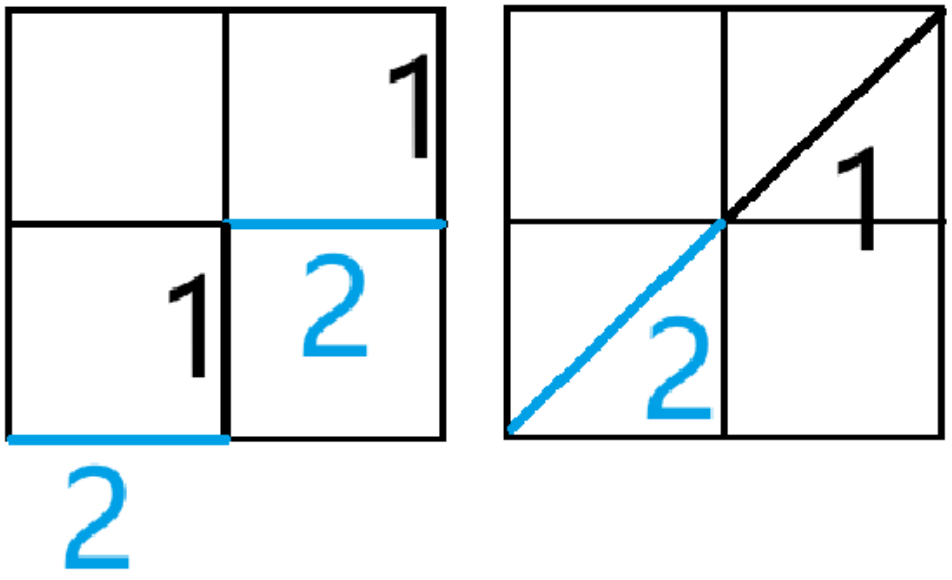
> 输出
>
> > 输出“first!”或者“second!”(不输出引号)，占一行。

这是一道数学题。
显然我们很容易得出两人博弈的点在于走对角线（可以用一步或两步）。

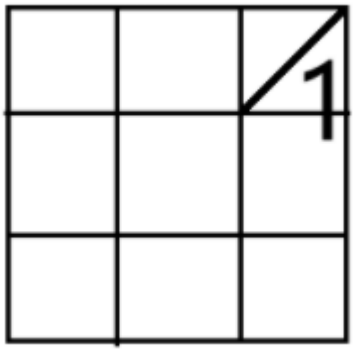


而我们观察到，任何$m \times n$的点阵总是能转换成一个正方形矩阵（边长为m,n中小的那个），因此，我们先考虑$m = n$这个特殊情况。

对于上图2×2的点阵中，显然，first走红线，就能胜利。



对于3×3的点阵中，显然first怎么走，都是输。

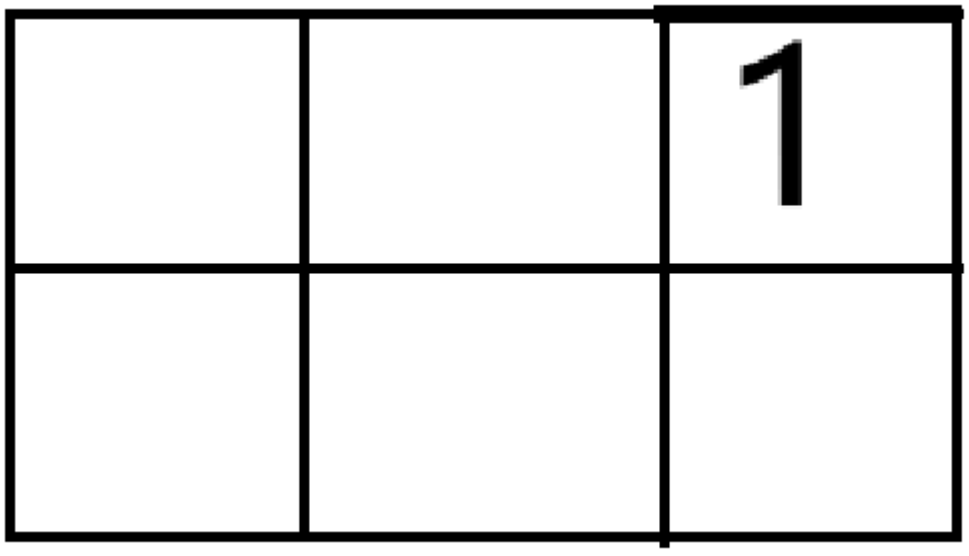


对于4×4的点阵中，显然，只需要这么走一步，first必赢。

通过这些，我们观察到，边长为奇数的正方形矩阵first赢，边长为偶数的正方形矩阵first输

根据以上结论，显然我们可以得到m-n能整除2时，总能构成与上述相同的正方形点阵。

现在考虑一下m-n不能整除2的情形：



显然，任意满足该条件的m,n，都能一步转换为一个边长为偶数的正方形矩阵，因此，first赢。

综上所有判定条件，可以得到 `m%2==0||n%2==0` 时，first赢，其余first输。

```c
#include <stdio.h>
int main() {
    int m, n;
    scanf("%d%d", &m, &n);
    if(n%2==0||m%2==0)printf("first!");
    else printf("second!");
}
```

拓展：求不同m,n对应的总方案数（动态规划）

```c
#include <stdio.h>
int main() {
    int m, n, i;
    int dp[201][201] = {0};
    scanf("%d %d", &m, &n);
    dp[1][1] = 1;
    for (i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (i == 1 && j == 1) continue;
            dp[i][j] = dp[i][j-1] + dp[i-1][j] + dp[i-1][j-
1];
        }
    }
    printf("%d\n", dp[m][n]);
}
```

5.明明的随机数

> 明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数（N <= 100），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

输入

> 输入包含两行：
> 第一行为1个正整数，表示所生成的随机数的个数N。
> 第二行有N个用空格隔开的正整数，为所产生的随机数。

输出

> 输出两行，第一行为1个正整数M，表示不相同的随机数的个数。第二行为M个用空格隔开的正整数，为从小到大排好序的不相同的随机数。

```c
#include <stdio.h>
void heapify(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}

void heap_sort(int arr[], int n) {
    int i;
    for (i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
```

```
    for (i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}

int main() {
    int N, random[100],i,M = 1;
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &random[i]);
    }
    heap_sort(random, N);
    for (i = 1; i < N; i++) {
        if (random[i] != random[i - 1]) {
            M++;
        }
    }
    printf("%d\n", M);
    printf("%d", random[0]);
    for (i = 1; i < N; i++) {
        if (random[i] != random[i - 1]) {
            printf(" %d", random[i]);
        }
    }
}
```

---

6. 8.2_1 正数个数填空

> 阅读下面程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。
> 下面`PositiveNum()`函数用于统计n个整数中正数的个数。

```
>#include <stdio.h>
int PositiveNum(int a[], int n)
{
    int i, count   ①   ;
    for(i = 0; i < n; i++)
    {
        if(a[i] > 0)    ②   ;
    }
    return    ③   ;
}
int main()
{
    int i, n, arr[20];
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("%d", PositiveNum(arr, n));
    return 0;
}
```

注意：请务必提交完整的程序代码，不要修改代码框架。

输入

> 输入包含二行：
> 第一行是整数n（0 < n <= 20）。
> 第二行有n个整数，相邻两项之间用一个空格隔开。

输出

> 输出正数的个数。

```c
#include <stdio.h>
int PositiveNum(int a[], int n)
{
    int i, count=0 ;
    for(i = 0; i < n; i++)
    {
        if(a[i] > 0) count++ ;
    }
    return count;
}
int main()
{
    int i, n, arr[20];
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("%d", PositiveNum(arr, n));
    return 0;
}
```

---

7. 8.2_2 计算数列的值填空

> 阅读下面程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。
> 下面Fib()函数使用迭代法计算Fibonacci数列前n项的值。

```c
#include <stdio.h>
void Fib(long f[],    ①    )
{
int i;
f[0] = 0;
f[1] = 1;
for(i = 2; i < n; i++)
{
```

```
f[i] =    ②   ;
}
}
int main()
{
    int i, n;
    long arr[30];
    scanf("%d", &n);
    Fib(arr, n);
    for(i = 0; i < n; i++)
    {
        printf("%ld ", arr[i]);
    }
    return 0;
}
```

注意：请务必提交完整的程序代码，不要修改代码框架。

输入

> 输入一个整数n（0 < n <= 30）。

输出

> 输出数列前n项的值，相邻两项之间用一个空格隔开。

```
#include <stdio.h>
void Fib(long f[], int n)
{
    int i;
    f[0] = 0;
    f[1] = 1;
    for(i = 2; i < n; i++)
    {
        f[i] = f[i - 1] + f[i - 2];
```

```
    }
}
int main()
{
    int i, n;
    long arr[30];
    scanf("%d", &n);
    Fib(arr, n);
    for(i = 0; i < n; i++)
    {
        printf("%ld ", arr[i]);
    }
    return 0;
}
```

---

8. 8.2_4 矩阵相乘填空

> 阅读下面程序，按要求在空白处填写适当的表达式或语句，使程序完整并符合题目要求。
> 利用矩阵相乘公式
>
> $$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{k,j}$$
>
> 编程计算m×n阶矩阵A和n×m阶矩阵B之积。

```
#include <stdio.h>
#define ROW 2
#define COL 3
/* 函数功能：计算矩阵相乘之积，结果存于二维数组c中 */void
MultiplyMatrix(int a[ROW][COL], int b[COL][ROW], int     ①
)
{
int i, j, k;
for(i = 0; i < ROW; i++)
{
```

```
for(j = 0; j < ROW; j++)
{
c[i][j] =    ②   ;
for(k = 0; k < COL; k++)
{
c[i][j] =    ③   ;
}
}
}
}
/* 函数功能：输出矩阵a中的元素 */void PrintMatrix(int a[ROW]
[ROW])
{
    int i, j ;
    for(i = 0; i < ROW; i++)
    {
        for(j = 0; j < ROW; j++)
        {
            printf("%6d", a[i][j]);
        }
        ④   ;
    }
}
int main()
{
    int a[ROW][COL], b[COL][ROW], c[ROW][ROW], i, j;
    printf("Input 2*3 matrix a:\n");
    for(i = 0; i < ROW ;i++)
    {
        for(j = 0; j < COL; j++)
        {
            scanf("%d",    ⑤   );
        }
    }
    printf("Input 3*2 matrix b:\n");
    for(i = 0; i < COL; i++)
    {
        for(j = 0; j < ROW; j++)
        {
            scanf("%d",    ⑥   );
```

```
        }
    }
    MultiplyMatrix(    ⑦    );
    printf("Results:\n");
    PrintMatrix(c);
    return 0;
}
```

> 注意：请务必提交完整的程序代码，不要修改代码框架。

答案

```
#include <stdio.h>
#define ROW 2
#define COL 3
/* 函数功能：计算矩阵相乘之积，结果存于二维数组c中 */void
MultiplyMatrix(int a[ROW][COL], int b[COL][ROW], int c[ROW]
[ROW])
{
    int i, j, k;
    for(i = 0; i < ROW; i++)
    {
        for(j = 0; j < ROW; j++)
        {
            c[i][j] = 0;
            for(k = 0; k < COL; k++)
            {
                c[i][j] = c[i][j] + a[i][k] * b[k][j];
            }
        }
    }
}
/* 函数功能：输出矩阵a中的元素 */void PrintMatrix(int a[ROW]
[ROW])
{
    int i, j ;
    for(i = 0; i < ROW; i++)
    {
        for(j = 0; j < ROW; j++)
```

```
        {
            printf("%6d", a[i][j]);
        }
        printf("\n");
    }
}
int main()
{
    int a[ROW][COL], b[COL][ROW], c[ROW][ROW], i, j;
    printf("Input 2*3 matrix a:\n");
    for(i = 0; i < ROW ;i++)
    {
        for(j = 0; j < COL; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Input 3*2 matrix b:\n");
    for(i = 0; i < COL; i++)
    {
        for(j = 0; j < ROW; j++)
        {
            scanf("%d", &b[i][j]);
        }
    }
    MultiplyMatrix(a, b, c);
    printf("Results:\n");
    PrintMatrix(c);
    return 0;
}
```

---

9. 8.3 数组元素相除改错

> 分析下面 `DivArray()` 函数能否实现“返回一个数组中所有元素被第一个元素除的结果”的功能？代码中存在怎样的错误隐患？请不要改动for循环体内语句，也不要使用新变量，编写出正确的程序。

```c
#include <stdio.h>
void DivArray(int *pArray, int n)
{
    int i;
    for(i = 0; i < n; i++)
    {
        pArray[i] /= pArray[0];
    }
}
int main()
{
    int i, n = 5, arr[5];
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    DivArray(arr, n);
    for(i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

> 注意：请务必提交完整的程序代码，不要修改代码框架。

输入

> 输入5个整数，相邻两项之间用一个空格隔开。

输出

> 输出 `DivArray()` 函数执行完毕后，数组各个元素的值。相邻两项之间用一个空格隔开。

```c
#include <stdio.h>
void DivArray(int *pArray, int n)
{
    int i;
    for(i = n-1; i >= 0; i--)
    {
        pArray[i] /= pArray[0];
    }
}
int main()
{
    int i, n = 5, arr[5];
    for(i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    DivArray(arr, n);
    for(i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

---

10. 8.5 成绩与平均分问题

> 编写函数`ReadScore()`和`Average()`，输入某班学生某门课的成绩（最多不超过40人），当输入为负值时，表示输入结束，用函数编程统计成绩高于平均分的学生人数。
>
> ---
>
> 输入
>
> > 输入若干个整数，相邻两项之间用一个空格隔开。

输出

> 输出成绩高于平均分的学生人数。

```
#include <stdio.h>
void ReadScore(int *pScore){
    int i;
    for(i = 0; i < 41; i++)
    {
        scanf("%d", &pScore[i]);
        if(pScore[i] < 0) break;
    }
}
void Average(int *pScore){
    int sum = 0,n = 0,i,count = 0;
    float average;
    for(i = 0; i < 41; i++){
        if(pScore[i] < 0) break;
        else{
            sum += pScore[i];
            n++;
        }
    }
    average = (float)sum / n;
    for(i = 0; i < n; i++){
        if(pScore[i] > average) count++;
    }
    printf("%d", count);
}
int main(){
    int score[41];
    ReadScore(score);
    Average(score);
}
```