LESSON_08 参考试题

一、 编程题

1. 填幻方（样题）

【问题描述】

在一个 N×N 的正方形网格中，每个格子分别填上从 1 到 N×N 的正整数，使得正方形中任一行、任一列及对角线的几个数之和都相等，则这种正方形图案就称为“幻方”（输出样例中展示了一个 3×3 的幻方）。我国古代称为“河图”、“洛书”，又叫“纵横图”。

幻方看似神奇，但当 N 为奇数时有很方便的填法：

1) 一开始正方形中没有填任何数字。首先，在第一行的正中央填上 1。

2) 从上次填数字的位置向上移动一格，如果已经在第一行，则移到同一列的最后一行；再向右移动一格，如果已经在最右一列，则移动至同一行的第一列。如果移动后的位置没有填数字，则把上次填写的数字的下一个数字填到这个位置。

3) 如果第 2 步填写失败，则从上次填数字的位置向下移动一格，如果已经在最下一行，则移到同一列的第一行。这个位置一定是空的（这可太神奇了!），把上次填写的数字的下一个数字填到这个位置。

4) 重复 2、3 步骤，直到所有格子都被填满，幻方就完成了!

快来编写一个程序，按上述规则，制作一个 N×N 的幻方吧。

【输入描述】

输入为一个正奇数 N，保证 3≤N≤21。

【输出描述】

输出 N 行，每行 N 个空格分隔的正整数，内容为 N×N 的幻方。

【样例输入】

3

【样例输出】

8 1 6

3 5 7

4 9 2

【参考代码】

```
#include<iostream>
using namespace std;
int main() {
    int n = 0;
    cin>>n;
    int cube[21][21]={0};
    int x=0, y=n/2;
    cube[x][y] = 1; // 第 1 步，第一行正中填写 1
    for (int d = 2;d<=n*n;d++) {
        int nx = (x+n-1)%n;
        int ny = (y+1)%n; // 第 2 步，向右上移动一格
        if (cube[nx][ny]!=0) {
            nx = (x+1)%n; // 第 3 步，如果第 2 步失败，向下移动一格
            ny = y;
        }
        cube[nx][ny] = d; // 填写下一个数字
        x = nx;
        y = ny;
    }
    //输出幻方
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++)
            cout<<cube[i][j]<<" ";
        cout<<endl;
    }
    return 0;
}
```

2. 寻找鞍点

【问题描述】

求一个 n 行 m 列矩阵的鞍点。鞍点指的是矩阵中的一个元素，它是所在行的最大值，并且是所在列的最小值。

例如下面的 5*5 的整数矩阵中，第 4 行第 1 列的元素就是鞍点，值为 8。

11	3	5	6	9
12	4	7	8	10
10	5	6	9	11
8	6	4	7	2
15	10	11	20	25

编程输入一个 n 行 m 列矩阵，寻找矩阵中的鞍点，输出鞍点的位置，如果不存在鞍点，则输出"none"。

【输入描述】

共 n+1 行;
第一行两个正整数 n 和 m，1≤n、m≤100，行号列号均从 1 开始;
接下来 n 行 m 列的整数矩阵，数据之间使用空格隔开。
【输出描述】
如果存在鞍点，输出鞍点的行列号和值，三个数据之间用空格隔开。
如果不存在鞍点，输出字符串"none"。
【样例输入 1】
5 5
11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25
【样例输出 1】
4 1 8
【样例输入 2】
2 3
1 2 3
4 5 2
【样例输出 2】
none
【参考代码】

```cpp
#include <iostream>
using namespace std;
int a[100][100];
int main(){
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>a[i][j];
    int x=0,y=0;// 记录鞍点位置
    bool f=true;// 标记是否找到鞍点
    for(int i=1;i<=n;i++){
        f=true;// 假设每次能找到
        x=i;y=1;
        // 先在每行找最大
        for(int j=1;j<=m;j++)
            if(a[i][j]>a[x][y])y=j;
        // 再判断 a[x][y]是否为所在列最小
        for(int h=1;h<=n;h++){
            if(a[h][y]<a[x][y]){
                f=false;
                break;
            }
        }
        if(f)break;// 鞍点是唯一的，找到鞍点退出整个循环
    }
    if(f)cout<<x<<" "<<y<<" "<<a[x][y];
    else cout<<"none";
    return 0;
}
```