一、编程题

ACM：图像物体的边界

题目描述：给定一个二维数组M行N列，二维数组里的数字代表图片的像素，为了简化问题，仅包含像素1和5两种像素，每种像素代表一个物体，2个物体相邻的格子为边界，求像素1代表的物体的边界个数。
像素1代表的物体的边界指与像素5相邻的像素1的格子，边界相邻的属于同一个边界，相邻需要考虑8个方向（上，下，左，右，左上，左下，右上，右下）。
其他约束：
地图规格约束为：
0<M<100
0<N<100

1）如下图，与像素5的格子相邻的像素1的格子（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）、（4,4）、（4,5）、（5,4）为边界，另（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）相邻，为1个边界，（4,4）、（4,5）、（5,4）相邻，为1个边界，所以下图边界个数为2。

1	1	1	1	1	1
1	5	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	5

2）如下图，与像素5的格子相邻的像素1的格子（0,0）、（0,1）、（0,2）、（1,0）、（1,2）、（2,0）、（2,1）、（2,2）、（3,3）、（3,4）、（3,5）、（4,3）、（4,5）、（5,3）、（5,4）、（5,5）为边界，另这些边界相邻，所以下图边界个数为1。注：（2,2）、（3,3）相邻。

1	1	1	1	1	1
1	5	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	5	1
1	1	1	1	1	1

输入描述：
第一行，行数 M，列数 N

第二行开始，是 M 行 N 列的像素的二维数组，仅包含像素 1 和 5

输出描述：
像素 1 代表的物体的边界个数。如果没有边界输出 0（比如只存在像素 1，或者只存在像素 5）。

补充说明：

示例1

输入：6 6

```
1 1 1 1 1 1
1 5 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 5
```

输出：2

说明：参考题目描述部分

示例2

输入：6 6

```
1 1 1 1 1 1
1 5 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 5 1
1 1 1 1 1 1
```

输出：1

说明：参考题目描述部分

代码：

```
#include <exception>
#include <iostream>
#include <vector>
using namespace std;

bool fs(int i, int j, vector<vector<int>> &ca, int &count)
{
	if (ca[i][j] == 0)
		return false;

	ca[i][j] = 0;
	fs(i+1, j+1, ca, count);
	fs(i+1, j-1, ca, count);
	fs(i-1, j+1, ca, count);
	fs(i-1, j-1, ca, count);
	fs(i, j+1, ca, count);
	fs(i, j-1, ca, count);
	fs(i+1, j, ca, count);
	fs(i-1, j, ca, count);

	// count++;
	return true;
}
```

```
int main() {
    int n, m, a;
    cin >> m >> n;
    m+=2;
    n+=2;
    auto vec = vector<vector<int>>(m, vector<int>(n));
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == 0 || i == m - 1 || j == 0 || j == n - 1)
            {
                vec[i][j] = 0;
            }
            else
            {
                cin >> a;
                vec[i][j] = a;
            }
            // cout << vec[i][j];
        }
    }

    auto ca = vector<vector<int>>(m, vector<int>(n, 0));
    // 保存边界
    for (int i = 1; i < m - 1; i++)
    {
        for (int j = 1; j < n - 1; j++)
        {
            if (vec[i][j] == 5)
                continue;
            if (vec[i+1][j+1] == 5 ||vec[i+1][j-1] == 5 ||vec[i-1][j+1] == 5 ||vec[i-1][j-1] == 5 ||
            vec[i+1][j] == 5 ||vec[i][j+1] == 5 ||vec[i-1][j] == 5 ||vec[i][j-1] == 5 )
            {
                ca[i][j] = 1;
            }
        }
    }

    // 遍历边界
    int count = 0;
    int count0 = 0;
    for (int i = 1; i < m - 1; i++)
```

```
	{
		for (int j = 1; j < n - 1; j++)
		{
			if (fs(i, j, ca, count))
				count0++;
		}
	}

	cout << count0 << endl;
	return 0;
}
```