题目描述：

学校组织活动，将学生排成一个矩形方阵。请在矩形方阵中找到最大的位置相连的男生数量。这个相连位置在一个直线上，方向可以是水平的、垂直的、呈对角线的或者反对角线的。

注：学生个数不会超过 10000.

输入描述：

输入的第一行为矩阵的行数和列数，接下来的 n 行为矩阵元素，元素间用“,”分隔。

输出描述：

输出一个整数，表示矩阵中最长的位置相连的男生个数。

示例 1

输入：

```
3,4
F,M,M,F
F,M,M,F
F,F,F,M
```

输出：

```
3
```

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;


bool inAear(int m, int n, int i, int j){
	if (i < 0 || i >= m || j < 0 || j >= n) {
		return false;
	}
	return true;
}
int MyGetMaxNum(vector<vector<char>> grids,int m,int n, int i, int j) {
	int res = 0;
	vector<int> a(4, 1);
	//上
	for (int x = i - 1; x >= 0; x--) {
		if (grids[x][j] == 'M') {
			a[0]++;
		}
		else {
```

```
                break;
        }
}
//下
for (int x = i + 1; x < m; x++) {
        if (grids[x][j] == 'M') {
                a[0]++;
        }
        else {
                break;
        }
}
//右
for (int y = j + 1; y < n; y++) {
        if (grids[i][y] == 'M') {
                a[1]++;
        }
        else {
                break;
        }
}
//左
for (int y = j- 1; y >=0 ; y--) {
        if (grids[i][y] == 'M') {
                a[1]++;
        }
        else {
                break;
        }
}
//斜
for (int x = i - 1, y = j - 1; inAear(m,n,x, y); x--, y--) {
        if (grids[x][y] == 'M') {
                a[2]++;
        }
        else {
                break;
        }
}
for (int x = i + 1, y = j + 1; inAear(m, n, x, y); x++ , y++) {
        if (grids[x][y] == 'M') {
                a[2]++;
        }
        else {
```

```
                break;
            }
        }
        for (int x = i - 1, y = j + 1; inAear(m, n, x, y); x--, y++) {
            if (grids[x][y] == 'M') {
                a[3]++;
            }
            else {
                break;
            }
        }
        for (int x = i + 1, y = j - 1; inAear(m, n, x, y); x++, y--) {
            if (grids[x][y] == 'M') {
                a[3]++;
            }
            else {
                break;
            }
        }
        res = *max_element(a.begin(), a.end());
        return res;




}


int main() {
    string tmp;
    while (getline(cin,tmp)) {
        vector<vector<char>> grids;//(m, vector<char>(n))
        int found = tmp.find(",");
        int m = stoi(tmp.substr(0, found));
        int n = stoi(tmp.substr(found+1));
        int k = m;
        while (k--) {
            getline(cin, tmp);
            vector<char> qwe;
            for (auto x : tmp) {
                if (x == ',') {
                    continue;
                }
                else {
```

```
                qwe.push_back(x);
            }
        }
        grids.push_back(qwe);
    }

    int res = 0;
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (grids[i][j] == 'M') {
                res = max(res, MyGetMaxNum(grids, m, n, i, j));
            }
        }
    }
    std::cout << res;
}


    return 0;
}
```