Game Rules

R = robber, C = cop, N = number of rows, K = number of columns, m = multiplicity

1. Robbers are placed
2. Cops are placed
3. Venerable cops are removed
4. Robbers move
5. Venerable robbers are removed
6. Cops move
7. Venerable cops are removed
8. Repeat from step 4, until either one of them is left on the board.

Other Rules:

1. An entity is said to be venerable when it is surrounded by more opponent entity then its own kind including itself.
2. Multiplicity is the number of cops on each grid containing a C.
3. If the game runs longer than certain number of moves it's declared as "unsolvable" and multiplicity is increased.
4. If no entity can move for its turn then all of them is declared venerable and removed in the next step, the situation is a "deadlock".
5. There must be at least one free block.

Minimum Multiplicity

```
int FindMinM(int r, int c)
{
	// 3
	if(r >= (2 * k) + 1) return 3;

	// 2
	if(r >= 8) return 2; // diagonal

	if(r == k && c > r && (c%k) <= (k+1)/2)  return 2;
	if(r >= k + 1) return 2;

	// 1
	return 1;
}
```

Case 1:

```
if(r >= (2 * k) + 1) return 3;
```

				R	R	R
				R	R	
				R	R	

Number of Robbers, r, is more than the twice the smaller side, f, the min multiplicity of cops required to break the formation is 3.

			CC C	**R**	R	R
			CC C	**R**	R	
			CC C	**R**	R	

The robbers **R** bold and big are vulnerable.

Case 2:

```
if(r == k && c > r && (c%k) <= (k+1)/2)  return 2;
```

C	R	C	C	C
R		C	C	C
C	R	C	C	C
	R	C	C	C
C	R	C	C	C

>

C	R	C	C	C
R		C	C	C
C	R	C	C	C
	R	C	C	C
C	R	C	C	C

>

	R	C	C	C
R		C	C	C
	R	C	C	C
	R	C	C	C
	R	C	C	C

>

	R	**C**	**C**	**C**
	R	**C**	**C**	**C**
	R	**C**	**C**	**C**
	R	**C**	**C**	**C**
	R	**C**	**C**	**C**

A typical case in which the robbers kill some of the initially placed cops and surround the rest in the next turn. None of the Cops can move and thus all of them are eliminated and the Robbers win.

Case 3:

```
if(r >= k + 1) return 2;
```

					R	R
					R	
					R	

Number of Robbers, r, is more than the smaller side, f, the min multiplicity of cops required to break the formation is 2.

				CC	**R**	R
				CC	**R**	
				CC	**R**	

The robbers **R** bold and big are vulnerable.

Case 4:

```
if(r >= 8) return 2; // diagonal
```

To make a diagonal formation, too many pegs are needed. But when the board is too big diagonal is the best formation using 8 pegs, the corner is kept to move freely.

CC						
R	CC					
R	R	CC				
	R	R	CC			
R		R	**R**	CC		

The robbers **R** (bold and big are vulnerable), for multiplicity = 2.

When the board is small the best formation is STRAIGHT, and for large board DIAGONAL is best.

Max Multiplicity

```
int mMax = min(r, 8) + 1; // (max number of robber that can surround a cop) + 1 (to overpower the robbers)
```

To find the max Multiplicity we consider one Cop surrounded by all robbers, as shown below. The max number of robbers surrounding a cop will be 8, so the max multiplicity for extreme case will be (8+1)=9.

		R	R	
		C	R	
			R	

Multiplicity

1. Find the **min multiplicity**.
2. Find the **worst arrangement** for the cops, for the particular multiplicity. To remove some cops in the first turn.
3. **Multiplicity** found
    a. In case the cops can be split into two groups and the number of cops is **more than double** the robbers
    b. Or if the cops are **more than** the robbers
4. Else increase the multiplicity until it reaches **max multiplicity**, go to step 2.