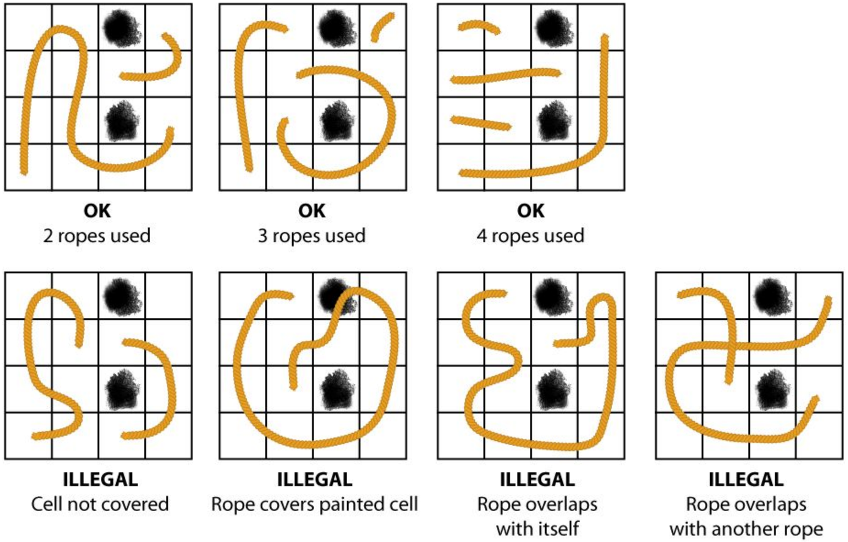
J182 - Rope

Percy Wong {percywtc}

The Problem



SCORING

ILLEGAL or **M > R + C + N**	0%
NICE (i.e. N + 1 < M ≤ R + C + N)	60%
EXCELLENT (i.e. M ≤ N + 1)	100%

SUBTASKS

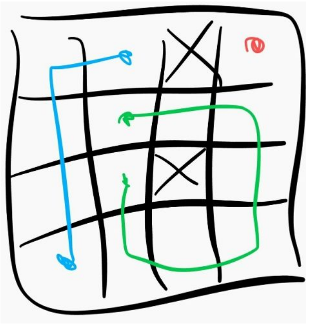
For all cases:

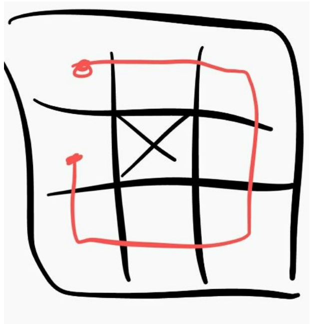
$1 \le R, C \le 300$

$0 \le N < R \times C$

	Points	Constraints
1	8	$R = C = 2$ $N = 0$
2	18	$R = 1$
3	21	$N = 0$
4	53	No additional constraints

Background





Problem Idea By - percywtc

Testdata By - percywtc; microtony

Initial version of this problem is a bit harder than the current version
This harder version will be discussed later on

Statistics

0 points	23 + 6 + 1 + 0 = 30
8 points	7 + 4 + 4 + 0 = 15
29 points	9 + 9 + 8 + 2 = 28
47 points	0 + 1 + 2 + 3 = 6
100 points	0 + 0 + 0 + 3 = 3

First solved by **mtyeung1** at **1h 17m 44s**

SUBTASKS

For all cases:

$1 \leq R, C \leq 300$

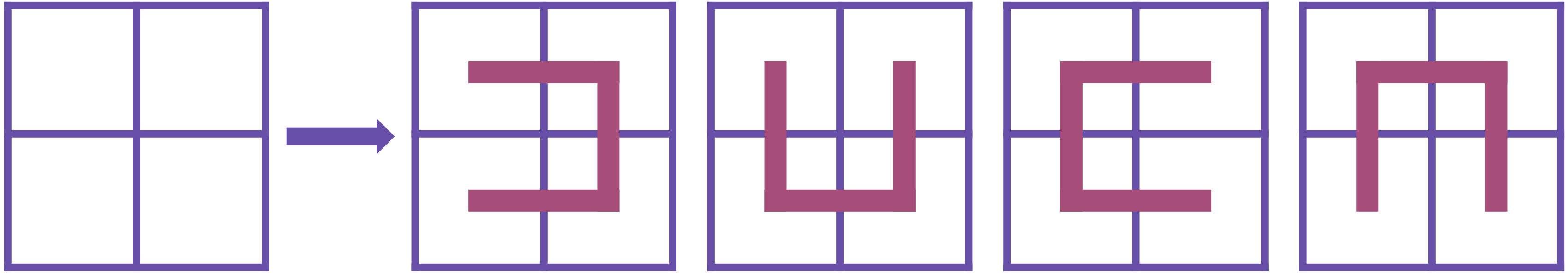
$0 \leq N < R \times C$

	Points	Constraints
1	8	$R = C = 2$ $N = 0$
2	18	$R = 1$
3	21	$N = 0$
4	53	No additional constraints

Solution 1 - The First Subtask

8 points for solving a single case - R = C = 2, N = 0

INPUT **OUTPUT**



```
2 2 0
```

```
1          1
4          4
1 1   OR   2 1
1 2        2 2
2 2        1 2
2 1        1 1
```

```
1          1
4          4
1 1   OR   1 2
2 1        2 2
2 2        2 1
1 2        1 1
```

```
1          1
4          4
1 2   OR   2 2
1 1        2 1
2 1        1 1
2 2        1 2
```

```
1          1
4          4
2 1   OR   2 2
1 1        1 2
1 2        1 1
2 2        2 1
```

Solution 1 - The First Subtask

This solution can only solve Subtask 1 with **M = 1** rope used

Subtask	Score	Max Score
1	8	8
2	0	18
3	0	21
4	0	53
Total	**8**	**100**

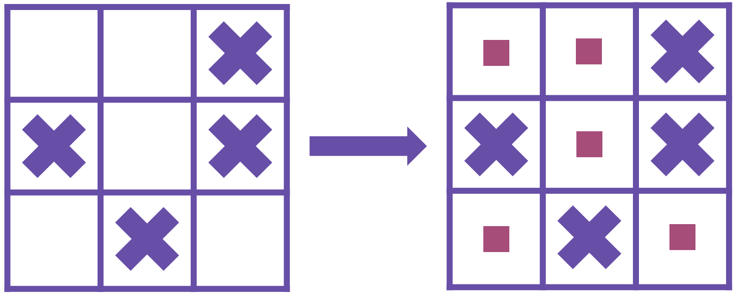
Solution 1 - The First Subtask

PSEUDOCODE

```
PrintLine(1)
PrintLine(4)
PrintLine(1 1)
PrintLine(1 2)
PrintLine(2 2)
PrintLine(2 1)
```

Solution 2 - Dots

Simply, for each unoccupied cell, place a rope

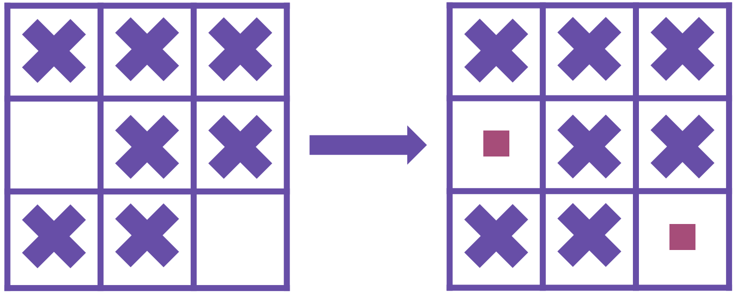


INPUT

```
3 3 4
1 3
2 1
2 3
3 2
```

OUTPUT

```
5
1
1 1
1
1 2
..
```



INPUT

```
3 3 7
1 1
1 2
1 3
2 2
..
```

OUTPUT

```
2
1
2 1
1
3 3
```

Solution 2 - Dots

In general, this solution needs **M = R × C - N** ropes

Worst case is when **N = 0**, which requires **M = R × C** ropes

For Subtask 1, not worse than **NICE** (i.e. M ≤ R + C + N) [4 ≤ 2 + 2 + 0]
For Subtask 2, not worse than **NICE** (i.e. M ≤ R + C + N) [C ≤ 1 + C + 0]
For Subtask 3 and 4, may reach M > R + C + N [R × C > R + C + 0]

Subtask	Score	Max Score
1	4.8	8
2	10.8	18
3	0	21
4	0	53
Total	**15.6**	**100**

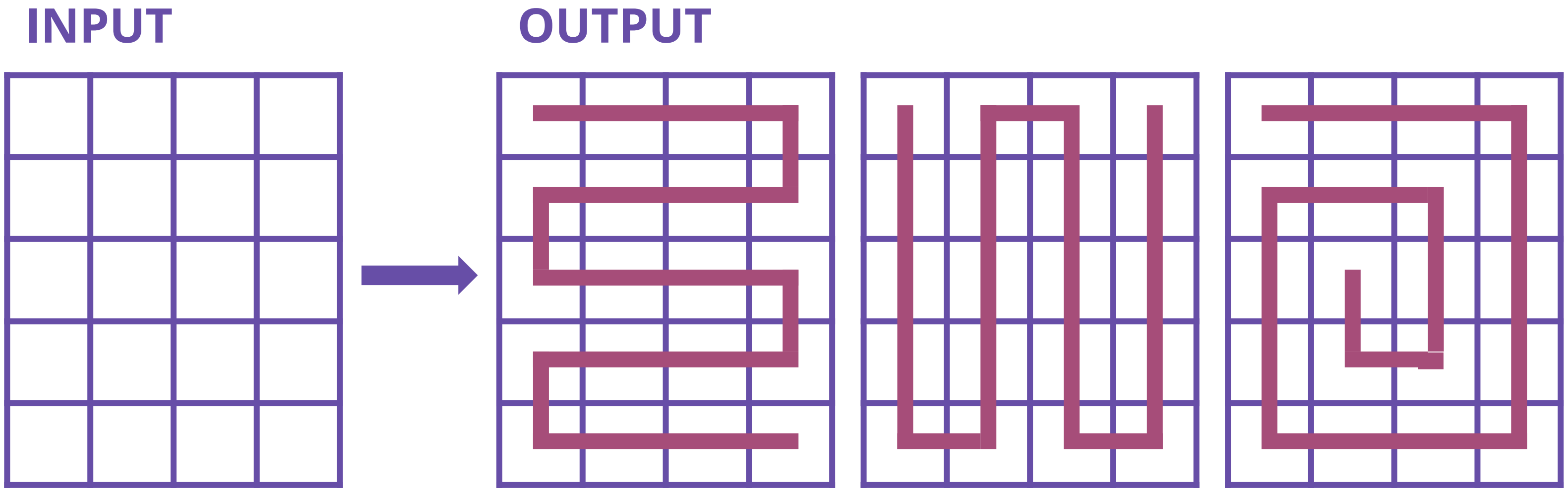
Solution 2 - Dots

PSEUDOCODE (For Subtask 2 Only)

```
For i = 1 .. N
  Read(x, y)
  A[y] = True
PrintLine(C - N)
For i = 1 .. C
  If (A[1][i] = False)
    PrintLine(1)
    PrintLine('1 ', i)
```

Solution 3 - Snake

For the entire grid empty (**N = 0** in Subtask 3),
We can only use **M = N + 1 = 1** rope to fill the whole grid



Solution 3 - Snake

This solution always need **M = 1** ropes,
And it is only suitable for solving **N = 0** cases

For Subtask 1 and 3, it is always an **EXCELLENT** placement (i.e. $M \le N + 1$)
This solution is not suitable for Subtask 2 and 4

Subtask	Score	Max Score
1	8	8
2	0	18
3	21	21
4	0	53
Total	**29**	**100**

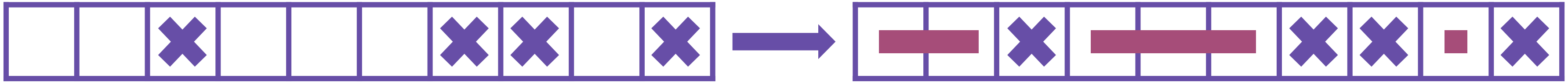
Solution 4 - Long Rope in Interval

For **R = 1** (Subtask 2),

optimal way is for each consecutive unoccupied interval, place a long rope

It is easy to see that this is the only optimal way

INPUT **OUTPUT**



INPUT:
```
1 10 4
1 3
1 7
1 8
1 10
```

OUTPUT:
```
3
2
1 1
1 2
3
1 4
1 5
1 6
1
1 9
```

Solution 4 - Long Rope in Interval

This solution needs at most **M = N + 1** ropes
as **N** obstacles can divide the row into no more than **N + 1** intervals

Worst case is when **N = 0**

For Subtask 2, must be **EXCELLENT** (i.e. M ≤ N + 1) [N + 1 ≤ N + 1]
This solution is not applicable to Subtask 1, 3 and 4

Subtask	Score	Max Score
1	0	8
2	18	18
3	0	21
4	0	53
Total	**18**	**100**

Solutions Summary

Solutions		1 - First Sub	2 - Dots	3 - Snakes	4 - Interval
Subtask	**Max Score**	**Score**			
1	8	8	4.8	8	0
2	18	0	10.8	0	18
3	21	0	0	21	0
4	53	0	0	0	0
Total	**100**	**8**	**15.6**	**29**	**18**

cumulative

Score
8
18
21
0
47

Solution 4 - Long Rope in Interval

PSEUDOCODE (For Subtask 2 Only)

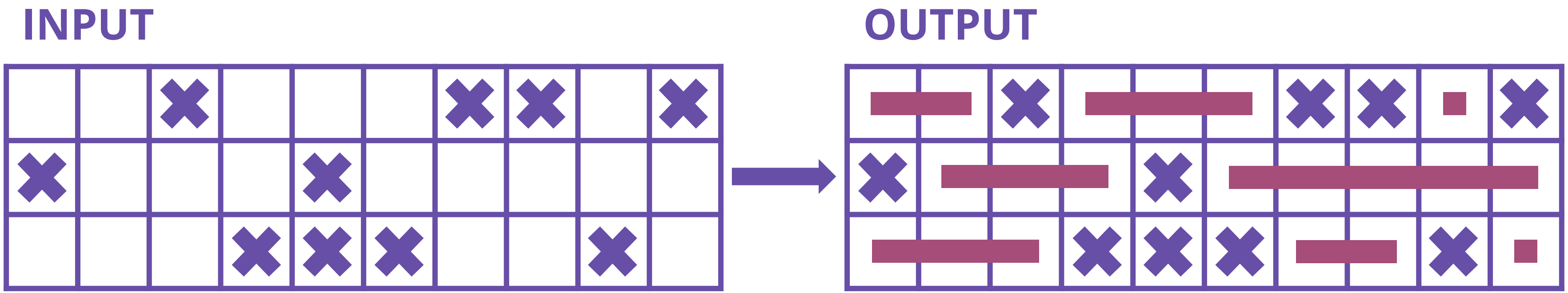
```
Count = 0
A[0] = A[C + 1] = True
For i = 0 .. C
  If (A[i] AND A[i+1])
    Count++
PrintLine(N + 1 - Count)
```

```
For i = 0 .. C
  If (A[i] AND NOT(A[i+1]))
    j = i + 1
    While (j<C+1 AND A[j]=False)
      j++
  PrintLine(j - i)
  For k = i .. j
    PrintLine('1 ', k)
  i = k + 1
```

Solution 5 - Copies of Solution 4

We can consider the each row separately

For each row, simply apply **Solution 4** once



Solution 5 - Copies of Solution 4

This solution needs at most **M = R + N** ropes

n obstacles can divide a row into no more than **n + 1** intervals
So, All **R** rows have in total no more than **N + R** intervals

For Subtask 2, must be **EXCELLENT** (i.e. $M \le N + 1$) [$N + 1 \le N + 1$]
For Subtask 1, 3 and 4, ast least **NICE** (i.e. $M \le R + C + N$) [$N + R \le R + C + N$]

Subtask	Score	Max Score
1	4.8	8
2	18	18
3	12.6	21
4	31.8	53
Total	**67.2**	**100**

Solutions Summary

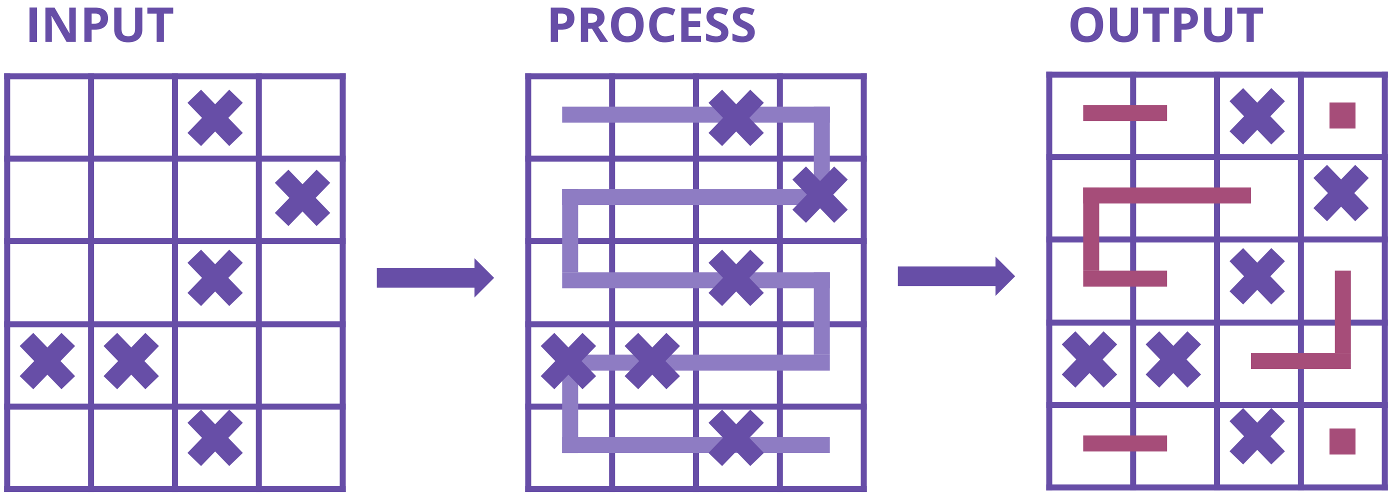
Solutions		1 - First Sub	2 - Dots	3 - Snakes	4 - Interval	5 - Copies
Subtask	Max Score	Score				
1	8	8	4.8	8	0	4.8
2	18	0	10.8	0	18	18
3	21	0	0	21	0	12.6
4	53	0	0	0	0	31.8
Total	**100**	**8**	**15.6**	**29**	**18**	**67.2**

cumulative →

Score
8
18
21
31.8
77.8

Solution 6 - Mix of Solution 3 & 4

We can consider the snake in **Solution 3** as a long line
Using **Solution 4 - Long Rope in Interval** on this long line



Solution 6 - Mix of Subtask 3 & 4

This solution considers a single long line with **N** obstacles
As mentioned in **Solution 4**, this requires at most **M = N + 1** ropes

For all Subtasks, it is always **EXCELLENT** (i.e. M ≤ N + 1) [N + 1 ≤ N + 1]

Subtask	Score	Max Score
1	8	8
2	18	18
3	21	21
4	53	53
Total	**100**	**100**

Summary

This task does not require you to solve every case with optimal arrangement,
But just “good enough” way

It is always a good idea to think of some small or special cases at first
It may lead you to develop a better solution

Other Solution - Depth First Search (DFS)

Choose an unoccupied cell,
Repeatedly walking to any unoccupied cell next to it until cannot do so
Repeat the whole process with another rope if there are still unoccupied cells

The score depends on your implementation:

Subtask	Score	Max Score
1	8	8
2	18	18
3	21	21
4	31.8	53
Total	**77.8**	**100**

Subtask	Score	Max Score
1	8	8
2	18	18
3	21	21
4	0	53
Total	**47**	**100**

{ WA on 7 out of 301 cases in Subtask 4

Harder Version of the Problem

CONSTRAINTS

R, C ≥ 2

R × C is even

N ≥ 1

Only **N** ropes can be used

Can you solve it? :)