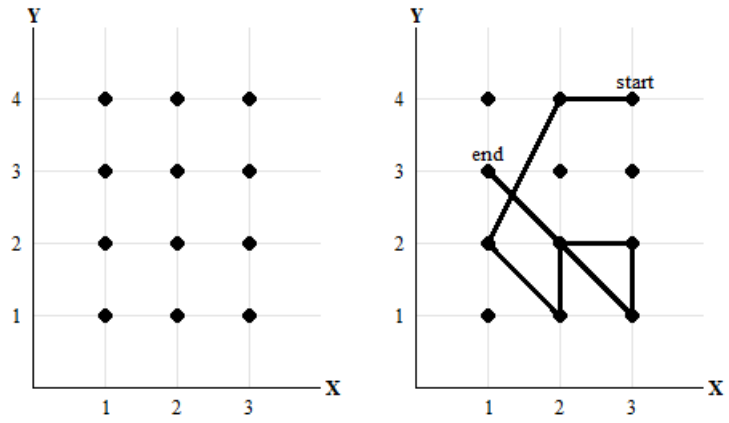
# Problem F. Lock Pattern

| | |
|---|---|
| Program: | `pattern.(cpp\|java)` |
| Input: | `pattern.in` |
| Balloon Color: | `Purple` |

One of the ways to lock some phones, is the lock pattern. To unlock your phone you have to draw a secret pattern on a grid of some points, the pattern will be some line segments connecting these points.

Your phone pattern grid consists of four rows with three points in each row. The following image on the left is the representation of the grid, it can be modeled as a 2D plane with $X$ and $Y$ coordinates for each point, for example the top left point is (1,4) and the bottom right point is (3,1). The image on the right is a valid pattern, which connects the following points in this order: (3,4) (2,4) (1,2) (2,1) (2,2) (3,2) (3,1) (1,3).




A valid pattern has the following properties:

1. A pattern can be represented using the sequence of points which it's touching for the first time (in the same order of drawing the pattern), a pattern going from (1,1) to (2,2) is not the same as a pattern going from (2,2) to (1,1).
2. For every two consecutive points $A$ and $B$ in the pattern representation, if the line segment connecting $A$ and $B$ passes through some other points, these points must be in the sequence also and comes before $A$ and $B$, otherwise the pattern will be invalid. For example a pattern representation which starts with (3,1) then (1,3) is invalid because the segment passes through (2,2) which didn't appear in the pattern representation before (3,1), and the correct representation for this pattern is (3,1) (2,2) (1,3). But the pattern (2,2) (3,2) (3,1) (1,3) is valid because (2,2) appeared before (3,1).
3. In the pattern representation we don't mention the same point more than once, even if the pattern will touch this point again through another valid segment, and each segment in the pattern must be going from a point to another point which the pattern didn't touch before and it might go through some points which already appeared in the pattern.
4. The length of a pattern is the sum of the Manhattan distances between every two consecutive points in the pattern representation. The Manhattan distance between two points $(X_1,Y_1)$ and $(X_2,Y_2)$ is $|X_1 - X_2| + |Y_1 - Y_2|$ (where $|X|$ means the absolute value of $X$). The length of the pattern in the above image is: $1 + 3 + 2 + 1 + 1 + 1 + 4 = 13$.
5. A pattern must touch at least two points.

Unfortunately you forgot your phone's pattern, but you remember the length of the pattern and a set of points $S$ which are not touched by the pattern for sure (the points which are not in $S$ might and might not be touched by the pattern), and you decided to try all the valid patterns which satisfy what you remember. Before doing this, you have to write a program to calculate for you the number of different valid patterns.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer $T$, the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of a test case contains two integers $L$ and $N$ separated by a single space. $L$ ($1 \leq L \leq 1{,}000$) is the length of the pattern (as described above), and $N$ ($0 \leq N \leq 12$) is the number of points that you are sure they are not touched by the pattern, followed by $N$ lines each line contains two integers $X$ ($1 \leq X \leq 3$) and $Y$ ($1 \leq Y \leq 4$) separated by a single space, representing the coordinates of one of the points which are not touched by the pattern for sure, the $N$ points are distinct.

## Output

For each test case, if there are no valid patterns according to what you remember, print on a single line "BAD MEMORY" (without the quotes), otherwise print the number of different valid patterns.

## Examples

| pattern.in | Standard Output |
|---|---|
| 3<br>1 0<br>2 10<br>1 1<br>1 2<br>1 3<br>2 1<br>2 2<br>2 3<br>2 4<br>3 2<br>3 3<br>3 4<br>1 3<br>1 4<br>2 4<br>3 4 | 34<br>BAD MEMORY<br>24 |