You are given input file containing K arrays, each consisting of N characters representing operations that can be performed on a movable object.
There are three possible operations:

- P – move one unit forward,
- L – rotate 90° counter-clockwise,
- D – rotate 90° clockwise.

An execution implies joining all arrays into single array of operations which, being applied to an initial position (x, y), will end in new position (x + Δx, y + Δy). When operations within the array must be executed in same order as they were originally located, it is allowed to arbitrarily reorder arrays.

Given that the object is initially positioned at (0, 0) facing north, find out number of possible executions that will finish in the same position i.e. (0, 0).

Input. In first line of input file, there are two integers K and N. In the next K lines, there will be N characters each representing single operation.
Output: In first and only line of output file, print number of possible reorderings which satisfy the goal.

Example:

Input	Output
3 4	1
PPLP	
PLPD	
LPLD	

Explanation. Given all possible reorderings:

- 1 - 2 - 3 [PPLP—PLPD—LPLD] ends in (-2, 0)
- 1 - 3 - 2 [PPLP—LPLD—PLPD] ends in (0, 0)
- 2 - 1 - 3 [PLPD—PPLP—LPLD] ends in (-2, 2)
- 2 - 3 - 1 [PLPD—LPLD—PPLP] ends in (-4, 0)
- 3 - 1 - 2 [LPLD—PPLP—PLPD] ends in (-2, -2)
- 3 - 2 - 1 [LPLD—PLPD—PPLP] ends in (-4, -2)

Only second reordering when arrays are executed in 1 - 3 - 2 will finish in position equal to the initial one.

Write an **efficient** algorithm for the following assumptions:

- K is an integer within the range [1..8]
- N is an integer within the range [1..100 000]
- Each element of array is one of {P, L, D}

Notes on performance
For all valid test cases, program must terminate in less than one second on average CPU and overall complexity must not be worse than O(n log n).