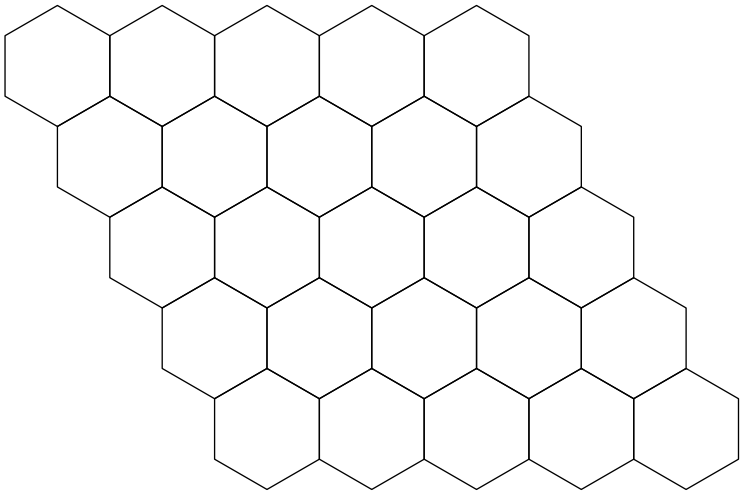
Problem D. Game of Hex

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	48 mebibytes

Note the unusually low memory limit and please read the note about Java below if you are using it.

The game of *Hex* is played on a rhombic hexagonal grid consisting of $n \times n$ hexagons. An example grid for $n = 5$ is pictured below:



Two players have their own colors, let's say red and blue. The players alternate turns, and in each turn a player colors a yet-uncolored hexagon with their own color. The red player starts, and the game ends when all hexagons have been colored.

The red player wins if the left and the right side of the board are connected by a path consisting only of red hexagons. The blue player wins if the top side and the bottom side of the board are connected by a path consisting only of blue hexagons. The beauty of the game lies in the fact that these conditions are mutually exclusive, and moreover there are no draws: in the end exactly one player always wins.

A path is a sequence of hexagons where each hexagon is adjacent (shares one of the six sides) to the previous one. Each of the four corner hexagons is considered connected to two sides of the board, in other words can be a starting/ending point of a winning path of either color.

You are given the current state of an ongoing game of Hex. Consider all fully colored boards that this game could end up with. In how many of them does the red player win?

Input

The first line of the input contains a single integer n, $2 \le n \le 12$.

The next n lines describe the board row-by-row, according to the picture above. Each line contains n characters. Each character is one of:

1. `.` (dot) denotes a yet-uncolored hexagon
2. `R` (capital letter R) denotes a hexagon colored red
3. `B` (capital letter B) denotes a hexagon colored blue

It is guaranteed that the given board is a valid intermediate state in the game of Hex, in other words that the number of red hexagons is either the same or one more than the number of blue hexagons.

Output

Print one integer: the number of final boards in which the red player wins. Note that we only count distinct final boards, the order of moves used to get there does not matter.

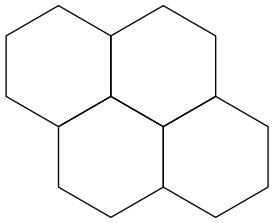
Examples

standard input
2 R. ..
standard output
1

standard input
12 ....
standard output
74010649922439309499690844774129439743805O

Note

In the first sample, the only final board where the red player wins looks like this:



In the second sample, since everything is symmetric the red player wins in exactly half of all $\binom{144}{72}$ possible final boards, and $\frac{1}{2} \cdot \binom{144}{72} = 74010649922439309499690844774129439743805O$.

If you submit this problem in Java, please use the special compilers "Oracle Java 8 (48ML)" and "Oracle Java 7 x32 (48ML)". These compilers execute your solution passing the flag `-Xmx48M` to the Java environment, making sure you have 48 MiB of heap available. The actual memory limit for those compilers is set to a larger value to allow for some JVM overhead. For all other languages, including other versions of Java, the memory limit is set to 48 MiB.