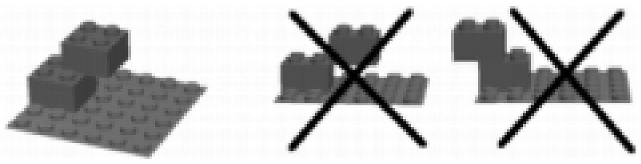
Lego

You are using Lego building blocks to train an artificial vision system. Write a program that, given pictures of a Lego construction taken from two angles, calculates in how many different ways it can be built.

In this task, there is only one kind of lego block (with 2 × 2 "knobs", see picture below), but it can have three different colors: white (W), gray (G) or black (B). All blocks exist in unlimited amounts. You use a quadratic base with 6 × 6 knobs. Every block must have its edges parallel to this base and no block may extend outside of it. Every block must rest upon at least one underlying block.


Left: An allowed way to place a block on top of another one. Center: An illegal way (the upper block hangs in the air). Right: Another illegal way (the upper block extends outside the base).

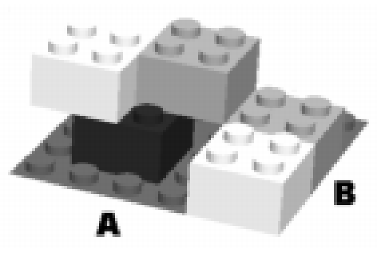
Input. The first line of the file `lego.in` contains H $(1 \leq H \leq 6)$, the height of the construction. Then follow H lines with 6 characters on each line, giving a picture of the construction as seen from one side (marked A on the figure below). The jth character on the ith line specifies what you see looking at the jth column from the left on the ith row from above. Each character may be one of 'W', 'G', 'B' or '.', specifying a color ('W', 'G', or 'B') or a hole ('.'). Note that you cannot estimate the depth, so a color seen in a certain position may either belong to a block near the front edge, or further back, provided no other block is blocking the sight.

The first picture is followed by another set of H lines with the construction seen from an angle where the observer has moved 90 degrees counterclockwise around the construction (marked B on the figure below).

Output. The program should output one line to the `lego.out` output file, containing a single integer: the number of different Lego constructions that satisfy the pictures given in the input. Note that even if two different possible constructions could be obtained from each other by rotating or mirroring, they both should be counted. For the given input, the answer will always fit in a signed 64-bit integer.

Sample.

lego.in	lego.out
`2`	`6`
`WWGG..`	
`.BB.WW`	
`.WGG..`	
`WWGG..`	



One of the possible constructions in the example.