Black and White

Original statement in spanish at http://www.dc.uba.ar/events/icpc/download/problems/taip2011-problems.pdf

The famous game Black and White is a solitaire which is played using a set of identical chips. Each chip has two faces with different colors. Surprisingly enough, those colors are black and white.

The game starts by placing **N** chips forming a single line. There exists an objective pattern which is a given sequence of **N** colors black or white. In a single move, the player can choose a group of consecutive chips and invert their color, in other words, for each chip in the group, the color which was facing up, is facing down and the one which was facing down is facing up. The game finishes when the facing up colors of the chips are equal to the objective pattern.

Barby has just discovered this game and soon she realized that you can always won by inverting each chip individually if needed. To make this game more challenging to her, she wanted to win in the least possible number of moves. Note that Barby just cares about how many moves she makes, and it doesn't matter how many chips are inverted in each move. To know how well is Barby playing, she asked you to make a program that given the chip's initial position and the objective pattern, shows the least possible number of moves to win the game. Are you going to say no?

Input

The input contains several test cases. Each test case is described in a single line that contains two non-empty words **S** and **T** of equal size and at most 500 letters each. **S** indicates the chip's initial position while **T** represents the objective pattern. Both words only contain uppercase letters "B" and "N", representing respectively White and Black. The last line of the input contains two asterisks separated by a single space and should not be processed as a test case.

Output

For each test case output a single line with an integer representing the least possible number of moves such that you can change the chip's which are initially positioned as described in **S** to form the pattern given by **T**.

Example

Input:

```
BBNBBNBBBB NNNNNBBNNB
BNBNB NBNBN
BNBN NBNB
B B
* *
```

Output:

```
3
```

1
1
0