Consider a problem of finding the best possible solution of a new kind of right alignment of two strings. As an example, here is the best such right alignment for two strings `AADDEFGGHC` and `ADCDEGH`:

```
AAD-DEFGGHC

 ADCDE--GH-
```

Each vertical matching position scores 2, and each continuous run of gaps scores -1. Thus the total score of an alignment is twice the number of matching positions, less the number of continuous runs of gaps. In the given example, six positions (`A`, `D`, `D`, `E`, `G`, `H`) match, and three runs of gaps are introduced. Thus the score of this alignment is $2 \times 6 + (-1) \times 3 = 9$. Note that we do not penalize for misalignment at the left, as we only consider the problem of finding the best right alignment.

You are to write a program that finds the score of the best right alignment for any two given strings.

Input

The first line of the input file contains an integer N ($1 \leq N \leq 10,000$) indicating the number of test cases to follow. Each test case consists of two lines each of which contains a string of length at most 50. The strings are composed entirely of alpha-numeric characters.

Output

For each test case in the input print a line containing the score of the best possible right alignment of the two given strings.

Sample Input

```
2
AADDEFGGHC
ADCDEGH
ERTTHCBYNQC
BEARTBCHQYN
```

Sample Output

```
9
8
```