Kick Start 2018 - Round F

Common Anagrams

Problem

Ayla has two strings **A** and **B**, each of length **L**, and each of which is made of uppercase English alphabet letters. She would like to know how many different substrings of **A** appear as anagrammatic substrings of **B**. More formally, she wants the number of different ordered tuples (i, j), with 0 ≤ i ≤ j < **L**, such that the i-th through j-th characters of **A** (inclusive) are the same multiset of characters as at least one contiguous substring of length (j - i + 1) in **B**.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with one line, containing **L**: the length of the string. The next two lines contain one string of **L** characters each: these are strings **A** and **B**, in that order.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the answer Ayla wants, as described above.

Limits

1 ≤ **T** ≤ 100.
Time limit: 20 seconds per test set.
Memory limit: 1 GB.
1 ≤ **L** ≤ 50.

Small dataset (Test set 1 - Visible)

The two strings **A** and **B** will consist only of the characters `A` and `B`.

Large dataset (Test set 2 - Hidden)

No additional constraints.

Sample

Sample Input

```
6
3
ABB
BAB
3
BAB
ABB
6
CATYYY
```

Sample Output

```
Case #1: 5
Case #2: 6
Case #3: 6
Case #4: 6
Case #5: 10
Case #6: 9
```

```
XXXTAC
9
SUBXXXXXX
SUBBUSUSB
4
AAAA
AAAA
19
PLEASEHELPIMTRAPPED
INAKICKSTARTFACTORY
```

In Sample Case #1, **L** = 3, **A** = `ABB`, and **B** = `BAB` There are 6 substrings of **A**:

- `A`. The substring `A` in **B** is (trivially) an anagram.
- `B`. The substring `B` in **B** is (trivially) an anagram.
- `B`. The substring `B` in **B** is (trivially) an anagram.
- `AB`. The substring `AB` in **B** is (trivially) an anagram.
- `BB`. There is no corresponding anagrammatic substring in **B**.
- `ABB`. The substring `BAB` in **B** is an anagram.

In total, there are 5 substrings with a corresponding anagrammatic substring in **B**, so the answer is 5.

In Sample Case #2, note that it is the same as Sample Case #1, except that **A** and **B** are swapped. This changes the answer to 6!

In Sample Case #3, note that the substring `CAT` in **A** has the corresponding substring `TAC` in **B** which is an anagram. This still counts, even though the strings are at different indices in their respective strings.

In Sample Case #4, note that although the substring `SUB` in **A** has several corresponding substrings in **B** which are anagrams, it only counts once.

In Sample Case #5, note that every substring of **A** has a corresponding anagrammatic substring in **B**, so the answer is 10.