

1A Compute the Number of Occurrences of a Pattern in a Text

Pattern Count Problem

Implement `PatternCount`.

Input: Strings *Text* and *Pattern*.

Output: The number of occurrences of *Pattern* in *Text*.

AGAGATCAGA
AGAGA AGA
1 2 3

Formatting

Input: Newline-separated strings *Text* and *Pattern*.

Output: An integer representing the number of times *Pattern* appears in *Text*.

Constraints

- The length of *Text* will be between 1 and 10^4 .
- The length of *Pattern* will be between 1 and 10^1 .
- *Text* and *Pattern* will be DNA strings.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code. Notice that `GCG` occurs twice in *Text*: once at the beginning (**GCG**CG) and once at the end (GCG**GCG**). A common mistake for this problem is incorrectly handling overlaps and not counting the second of these two occurrences (because it begins at the end of the previous occurrence).

Input:

GCGCG

GCG

Output:

2

Case 2

Description: This dataset just checks if you're correctly counting. It is the "easiest" test. Notice that all occurrences of `CG` in *Text* (ACGTACGTACGT) are away from the very edges (so your code won't fail on off-by-one errors at the beginning or at the end of *Text*) and that none of the occurrences of *Pattern* overlap (so your code won't fail if you fail to account for overlaps).

Input:

ACGTACGTACGT

CG

Output:

3

Case 3

Description: This dataset checks if your code correctly handles cases where there is an occurrence of *Pattern* at the very beginning of *Text*. Note that there are no overlapping occurrences of *Pattern* (i.e. `AAAA`), and there is no occurrence of *Pattern* at the very end of *Text*, so assuming your code passed Test Dataset 1, this test would only check for off-by-one errors at the beginning of *Text*.

Input:

AAAGAGTGTCTGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAATTTTATTGACTTAGG...
...TCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAATAATAATTACAGAGTACACAACATCCA
AAA

Output:

3

Case 4

Description: This dataset checks if your code correctly handles cases where there is an occurrence of *Pattern* at the very end of *Text*. Note that there are no overlapping occurrences of *Pattern* (i.e. AAAA), and there is no occurrence of *Pattern* at the very beginning of *Text*, so assuming your code passed Test Dataset 2, this test would only check for off-by-one errors at the end of *Text*.

Input:

```
AGCGTGCCGAAATATGCCGCCAGACCTGCTGCGGTGGCCTCGCCGACTTCACGGATGCCAAGTGCATAG...  
...AGGAAGCGAGCAAAGGTGGTTTCTTTGCTTTATCCAGCGCTTAACCACGTTCTGTGCCGACTTT  
TTT
```

Output:

4

Case 5

Description: This test dataset checks if your code is also counting occurrences of the Reverse Complement of *Pattern* (which would have an output of 4), which is out of the scope of this problem (that will come up later in the chapter). Your code should only be looking for perfect matches of *Pattern* in *Text* at this point.

Input:

```
GGACTTACTGACGTACG  
ACT
```

Output:

2

Case 6

Description: This dataset checks if your code correctly handles cases where occurrences of *Pattern* overlap. For example, any occurrence of the string CCC should count as 2 occurrences of CC (CCC and CCC). In this dataset, there are 5 occurrences of CC including overlaps. (ATCCGATCCCATGCCCATGTTG)

Input:

```
ATCCGATCCCATGCCCATG  
CC
```

Output:

5

Case 7

Description: This is the final test that we run your code on: the full dataset.

Input:

```
CTGTTTTTGATCCATGATATGTTATCTCTCCGTCATCAGAAGAACAGTGACGGATCGCCCTCTCTCTTG...  
...GTCAGGCGACCGTTTGCCATAATGCCCATGCTTCCAGCCAGCTCTCAAACCTCCGGTGACTCGCGC...  
...AGGTTGAGT  
CTC
```

Output:

9

Case 8

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.