Pattern Count

Input: Strings *Text* and *Pattern*
Output: *Count*(*Text*, *Pattern*)

Pseudocode

```
PatternCount(Text,Pattern)
    count ← 0
    for i ← 0 to |Text| - |Pattern|
        if Text(i,|Pattern|) = Pattern
            count ← count + 1
    return count
```

SAMPLE DATASET:

Input:

```
GCGCG

GCG
```

Output:

```
2
```

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 (GC**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). The sample dataset checks for the following things:

- Off-by-one at the beginning of Text (result would be 1)
- Off-by-one at the end of Text (result would be 1)
- Not counting overlaps (result would be 1)

TEST DATASET 1:

Input:

```
ACGTACGTACGT

CG
```

Output:

```
3
```

This dataset just checks if you're correctly counting. It is the "easiest" test. Notice that all occurrences of CG in Text (A**CG**TA**CG**TA**CG**T) 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).

TEST DATASET 2:

Input:

```
AAAGAGTGTCTGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAATTTTATTGAC
TTAGGTCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAATAATTACAGAGTA
CACAACATCCAT

AAA
```

Output:

```
4
```

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.

TEST DATASET 3:

Input:

```
AGCGTGCCGAAATATGCCGCCAGACCTGCTGCGGTGGCCTCGCCGACTTCACGGATGCCAAGTG
CATAGAGGAAGCGAGCAAAGGTGGTTTCTTTCGCTTTATCCAGCGCGTTAACCACGTTCTGTGC
CGACTTT

TTT
```

Output:

```
4
```

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.

TEST DATASET 4:
Input:
```
GGACTTACTGACGTACG

ACT
```
Output:
```
2
```

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.

TEST DATASET 5:

Input:

```
ATCCGATCCCATGCCCATG

CC
```

Output:

```
5
```

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" (**CC**C and C**CC**). In this dataset, there are 5 occurrences of CC including overlaps (AT**CC**GAT**CCC**ATG**CCC**ATG).

TEST DATASET 6:

Input:

```
CTGTTTTTGATCCATGATATGTTATCTCTCCGTCATCAGAAGAACAGTGACGGATCGCCCTCTC
TCTTGGTCAGGCGACCGTTTGCCATAATGCCCATGCTTTCCAGCCAGCTCTCAAACTCCGGTGA
CTCGCGCAGGTTGAGTA

CTC
```

Output:

```
9
```

This is the final test that we run your code on: the Full Dataset.