Pattern Matching Problem

Input: Two strings, *Pattern* and *Genome*

Output: All starting positions where *Pattern* appears as a substring of *Genome*

SAMPLE DATASET:

Input:

```
ATAT

GATATATGCATATACTT
```

Output:

```
1 3 9
```

The sample dataset is not actually run on your code.

TEST DATASET 1:

Input:

```
ACAC

TTTTACACTTTTTTGTGTAAAAA
```

Output:

```
4
```

This dataset checks if your code is written correctly but is also taking into account reverse complements, which we are not yet doing. Even though the reverse complement of “ACAC” (which is “GTGT”) occurs in Genome, we only want to count occurrences of “ACAC” specifically, which only occurs at index 4.

TEST DATASET 2:

Input:

```
AAA

AAAGAGTGTCTGATAGCAGCTTCTGAACTGGTTACCTGCCGTGAGTAAATTAAATTTTATTGAC
TTAGGTCACTAAATACTTTAACCAATATAGGCATAGCGCACAGACAGATAATAATTACAGAGTA
CACAACATCCAT
```

Output:

```
0 46 51 74
```

This dataset checks for off-by-one errors at the beginning of Genome. Notice that “AAA” occurs at the very beginning of Genome, so if you were to miss the first kmer of Genome, your code would output the following:

```
46    51    74
```

TEST DATASET 3:
Input:
```
TTT

AGCGTGCCGAAATATGCCGCCAGACCTGCTGCGGTGGCCTCGCCGACTTCACGGATGCCAAGTG
CATAGAGGAAGCGAGCAAAGGTGGTTTCTTTCGCTTTATCCAGCGCGTTAACCACGTTCTGTGC
CGACTTT
```
Output:
```
88 92 98 132
```

This dataset checks for off-by-one errors at the endof Genome. Notice that “TTT” occurs at the very end of Genome, so if you were to miss the last kmer of Genome, your code would output the following:

88 93 98

TEST DATASET 4:
Input:
```
ATA

ATATATA
```
Output:
```
0 2 4
```

This test dataset checks if your code correctly handles cases where instances of Pattern overlap in Genome. In this case, if you did not count overlaps, you would only find the first and last instances of ATA (**ATA**TATA and ATAT**ATA**). However, there is indeed a third occurrence, where the other two overlap (AT**ATA**TA).