2A Implement MotifEnumeration

Implanted Motif Problem
Find all (k, d)-motifs in a collection of strings.

Input: A collection of strings *Dna*, and integers *k* and *d*.
Output: All (*k*, *d*)-motifs in *Dna*.

```
AAAGGT
CCGCTA
TGGACT
AGTAAC
```

Formatting

Input: Space-separated integers *k* and *d*, followed by a space-separated collection of strings *Dna*.
Output: A space-separated list of strings representing all (*k*, *d*)-motifs in *Dna*.

Constraints

- The integer *k* will be between 1 and 10^1.
- The integer *d* will be between 1 and 10^1.
- The number of strings in *Dna* will be between 1 and 10^2.
- Each string in *Dna* will have a length between 1 and 10^2.
- All strings in *Dna* will be DNA strings.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
3 1
ATTTGGC TGCCTTA CGGTATC GAAAATT
```

Output:

```
ATA ATT GTT TTT
```

Case 2

Description: This dataset checks for off-by-one errors, both at the beginning and at the end. The 3-mers `ACG` and `CGT` both appear perfectly in all 3 strings in *Dna*. Thus, if your output doesn't contain `ACG`, you are most likely not counting the first *k*-mer of every string. Similarly, if your output doesn't contain `CGT`, you are most likely not counting the last *k*-mer of every string.

Input:

```
3 0
ACGT ACGT ACGT
```

Output:

```
ACG CGT
```

Case 3

Description: This dataset checks if your code work correctly when $d > 0$. If your code only counts motifs with $d = 0$ (and not $d > 0$), your code will only find a single motif (`AAA`, which is the only 3-mer that occurs perfectly in all of the strings of *Dna*). A correct solution would, in addition to `AAA`, find all 3-mers that differ from `AAA` by exactly 1 base.

Input:

```
3 1
AAAAA AAAAA AAAAA
```

Output:

```
AAA AAC AAG AAT ACA AGA ATA CAA GAA TAA
```

Case 4

Description: This dataset checks if your code counts motifs where the number of mismatches is equal to *d* in addition to motifs where the number of mismatches is less than *d*. For example, in this dataset, a correct solution would find *all* 3-mers (because we are allowing for 3 mismatches). However, an incorrect solution that counts mismatches less than *d* but not mismatches equal to *d* would only find the *k*-mers that differ from `AAA` by 1 or 2 bases, not the ones that differ from `AAA` by 3 bases.

Input:
```
3 3
AAAAA AAAAA AAAAA
```

Output:
```
AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA CAC CAG
CAT CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC
GCG GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG TCT TGA
TGC TGG TGT TTA TTC TTG TTT
```

Case 5

Description: This test dataset checks if your code is checking the last sequence in *Dna*. If your code only checks the first two sequences in the dataset, the 3-mer `AAA` exists perfectly in both and will thus be output. If your code checks the last sequence of *Dna* (`AACAA`), however, it will find that `AAA` does not appear. Thus, `AAA` is not a motif in *Dna*, and no sequences should be output.

Input:
```
3 0
AAAAA AAAAA AACAA
```

Output:

Case 6

Description: This test dataset checks if your code is checking the first sequence in *Dna*. If your code only checks the last two sequences in the dataset, the 3-mer `AAA` exists perfectly in both and will thus be output. If your code checks the first sequence of *Dna* (`AACAA`), however, it will find that `AAA` does not appear. Thus, `AAA` is not a motif in *Dna*, and no sequences should be output.

Input:
```
3 0
AACAA AAAAA AAAAA
```

Output:

Case 7

Description: A larger dataset of the same size as that provided by the randomized autograder.