Motif Enumeration

Input: Integers *k* and *d*, followed by a collection of strings *Dna*.
Output: All (*k*, *d*)-motifs in *Dna*

Pseudocode

```
MotifEnumeration(Dna,k,d)
    Patterns ← an empty set
    for each k-mer Pattern in Dna
        for each k-mer Pattern’ where d(Pattern’,Pattern) ≤ d
            if Pattern' appears in each string from Dna with at
            most d mismatches
                add Pattern' to Patterns
    remove duplicates from Patterns
    return Patterns
```

SAMPLE DATASET:

Input:

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

Output:

```
ATA ATT GTT TTT
```

The sample dataset is not actually run on your code.

TEST DATASET 1:

Input:

```
3 0
ACGT
ACGT
ACGT
```

Output:

```
ACG CGT
```

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.

TEST DATASET 2:

Input:

```
3 1
AAAAA
AAAAA
AAAAA
```

Output:

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

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 one base.

TEST DATASET 3:

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
```

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.

TEST DATASET 4:

Input:

```
3 0
AAAAA
AAAAA
AACAA
```

Output:

```
(nothing)
```

This test dataset checks if your code is checking the last sequence in Dna. If your code only checks sequences the first 2 sequences in the dataset, the 3-mer "AAA" exists perfectly in both and will thus be outputted. If your code checks the 3rd (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 outputted.

TEST DATASET 5:

Input:

```
3 0
AACAA
AAAAA
AAAAA
```

Output:

```
(nothing)
```

This test dataset checks if your code is checking the first sequence in Dna. If your code only checks sequences the last 2 sequences in the dataset, the 3-mer "AAA" exists perfectly in both and will thus be outputted. 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 outputted.