CSC 110 - Homework 9 - Bioinformatics

We have seen the code in class that finds all alignments between a pair of sequences by adding gaps to the shorter sequence. You can find a version of that code on the Sample Code page.

Of course, in a real bioinformatics application, gaps could occur in both sequences. For this assignment, you will modify the all_alignments program so that it allows the user to specify how many gaps to consider in the longer sequence. The program will find the optimal alignments between two sequences when a specified number of gaps is allowed in the longer sequence. For example, suppose we have the following two sequences:

ACT

GACG

If we allow for one gap in the longer sequence, then there are 5 possible sequences that result from adding that gap:

-GACG G-ACG GA-CG GAC-G GACG-

Then for each of these sequences, we would align it with the shorter sequence with 2 gaps, so that both sequences are the same length. There are 10 ways to add 2 gaps to ACT:

--ACT -A-CT -AC-T -ACT- A--CT

A-C-T A-CT- AC--T AC-T- ACT--

So the total number of alignments to consider in this example would be 5*10=50.

For this assignment, you will write a function:

```
scoreGapsInBoth(longSeq, shortSeq, longGaps):
      totalAlignments, highScore, highList
```

that computes the score for all alignments when given the two sequences, `longSeq` and `shortSeq`, and the number of gaps to allow in the long sequence (`longGaps`). From these scores, you will find the highest, and keep a list of all alignments with that score. The pseudocode for the function is as follows:

Find all sequences that result from adding the specified number of gaps to the longSeq
Find all sequences that result from adding enough gaps the the shorter sequence so that it is the same length as the longer sequence
For each of these long sequences with gaps
    For each of the short sequences with gaps
        Compute the score between the two sequences
        Keep track of the highest score and the list of sequences with that score

The function will return the total number of alignments, the high score, and the list of alignments with the high score.

You will also need to write a function

```
scoreAlignment(seq1, seq2):
    return score
```

That computes the score between the given sequences where we assume:

Gap = 0, Match = 1, Mismatch = -1

Skeleton code for this assignment can be found here:

hw10_skeleton.py

HINTS:

1) You will not need to make any changes to the functions in the given program. Your `scoreGapsInBoth` function will call the `insertAllGaps` function to compute the lists of sequences.
2) When you are storing the alignments with the optimal score, you will use a list of alignments. You can store each alignment as a concatenation of the two sequences in the alignment. For instance, if we have:

   `seq1 = 'AC-T-' and seq2 = '-GTCA'`

   then we can store the alignment as:

   `alignment = seq1 + '\n' + seq2`

   This will store the alignment as a single string, but will print with a line break so the two sequences are aligned under each other.

3) The main function is written for you, so all you have to write is the two functions described above. Write the `scoreAlignment` function first and test that it works correctly before writing the `scoreGapsInBoth` function.

Your program should NOT call the main function. Gradescope is expecting that the program will not execute until it calls the main function itself.

The output of your program should look like this:

```
>>> main()
Enter string 1: ACT
Enter string 2: GACG
How many total gaps should we allow in the longer sequence? 1
There are  50  alignments.
There are  2  optimal alignments with a score of  2 :
GAC-G
-ACT-

GACG-
-AC-T
```

```
>>> main()
Enter string 1: ACT
Enter string 2: GACG
How many total gaps should we allow in the longer sequence? 2
There are  300  alignments.
There are  12  optimal alignments with a score of  2 :
-GAC-G
--ACT-

-GACG-
--AC-T

G-AC-G
--ACT-

G-ACG-
--AC-T

GA-C-G
-A-CT-

GA-CG-
-A-C-T

GAC--G
-AC-T-

GAC--G
-ACT--

GAC-G-
-AC--T

GAC-G-
-ACT--

GACG--
-AC--T

GACG--
-AC-T-
```

```
>>> main()
Enter string 1: CCAGTCAGG
Enter string 2: AGTTTCCGAAT
How many total gaps should we allow in the longer sequence? 2
There are  55770  alignments.
There are  15  optimal alignments with a score of  3 :
--AGTTTCCGAAT
CCAG--T-C-AGG

--AGTTTCCGAAT
CCAG--TC--AGG

--AGTTTCCGAAT
CCAG--TCAG--G

--AGTTTCCGAAT
CCAG--TCAG-G-

--AGTTTCCGAAT
CCAG--TCAGG--

--AGTTTCCGAAT
CCAG-T--C-AGG

--AGTTTCCGAAT
CCAG-T-C--AGG

--AGTTTCCGAAT
CCAG-T-CAG--G

--AGTTTCCGAAT
CCAG-T-CAG-G-

--AGTTTCCGAAT
CCAG-T-CAGG--

--AGTTTCCGAAT
CCAGT---C-AGG

--AGTTTCCGAAT
CCAGT--C--AGG

--AGTTTCCGAAT
CCAGT--CAG--G

--AGTTTCCGAAT
CCAGT--CAG-G-

--AGTTTCCGAAT
CCAGT--CAGG--
```

```
>>> main()
Enter string 1: TACCTAGTCGGT
Enter string 2: TACCTAGTAGGTC
How many total gaps should we allow in the longer sequence? 1
There are  1274  alignments.
There are  2  optimal alignments with a score of  11 :
TACCTAGT-AGGTC
TACCTAGTC-GGT-

TACCTAGTA-GGTC
TACCTAGT-CGGT-


>>> main()
Enter string 1: TACCTAGTCGGT
Enter string 2: TACCTAGTAGGTC
How many total gaps should we allow in the longer sequence? 2
There are  47775  alignments.
There are  30  optimal alignments with a score of  11 :
-TACCTAGT-AGGTC
-TACCTAGTC-GGT-

-TACCTAGTA-GGTC
-TACCTAGT-CGGT-

T-ACCTAGT-AGGTC
T-ACCTAGTC-GGT-

T-ACCTAGTA-GGTC
T-ACCTAGT-CGGT-

TA-CCTAGT-AGGTC
TA-CCTAGTC-GGT-

TA-CCTAGTA-GGTC
TA-CCTAGT-CGGT-

TAC-CTAGT-AGGTC
TAC-CTAGTC-GGT-

TAC-CTAGTA-GGTC
TAC-CTAGT-CGGT-

TACC-TAGT-AGGTC
TACC-TAGTC-GGT-

TACC-TAGTA-GGTC
TACC-TAGT-CGGT-
```

```
TACCT-AGT-AGGTC
TACCT-AGTC-GGT-

TACCT-AGTA-GGTC
TACCT-AGT-CGGT-

TACCTA-GT-AGGTC
TACCTA-GTC-GGT-

TACCTA-GTA-GGTC
TACCTA-GT-CGGT-

TACCTAG-T-AGGTC
TACCTAG-TC-GGT-

TACCTAG-TA-GGTC
TACCTAG-T-CGGT-

TACCTAGT--AGGTC
TACCTAGT-C-GGT-

TACCTAGT--AGGTC
TACCTAGTC--GGT-

TACCTAGT-A-GGTC
TACCTAGT--CGGT-

TACCTAGT-A-GGTC
TACCTAGTC--GGT-

TACCTAGT-AG-GTC
TACCTAGTC-G-GT-

TACCTAGT-AGG-TC
TACCTAGTC-GG-T-

TACCTAGT-AGGT-C
TACCTAGTC-GGT--

TACCTAGT-AGGTC-
TACCTAGTC-GGT--

TACCTAGTA--GGTC
TACCTAGT--CGGT-

TACCTAGTA--GGTC
TACCTAGT-C-GGT-

TACCTAGTA-G-GTC
TACCTAGT-CG-GT-

TACCTAGTA-GG-TC
TACCTAGT-CGG-T-

TACCTAGTA-GGT-C
TACCTAGT-CGGT--

TACCTAGTA-GGTC-
TACCTAGT-CGGT--

>>>
```

Submit your program file to the corresponding assignment in Gradescope. Be sure to name the file `hw9.py`. Use the **Programming Rubric** to be sure you are including all elements of the program that are required.

NOTE: Gradescope is picky about function names and text matching exactly. So make sure that you use the function names described above and that your output looks exactly like the examples above.