CSC 110 – Lab 9
Bioinformatics Algorithms

Names: _____ Joey Koumjian _____

Lab Objectives:

- Write short functions manipulating and analyzing strings.
- Integrate functions into a larger program that is already written.
- Describe how the dot plot with sliding windows algorithm works.

Lab Policy:

- Please complete all sections of the lab, in order.
- Work with your partner using the rules of Pair Programming.
- When you get to an *Instructor Signature*, raise your hand. Do not move on until you get signed off.
- If you finish all required parts of the lab, move on to the Challenge Problems.
- If you finish all Challenge Problems, you may work on your homework that is due on Monday.
- You may not leave until the lab period is over.

Dot Plot with Sliding Windows

In class we looked at the simple dot plot algorithm. We discussed how it can be difficult to see patterns in the plot when we are matching each character individually. The sliding windows version of the dot plot allows us to visualize patterns in groups of characters.

In this lab you will implement the sliding windows algorithm for computing a dot plot.

I. Compute Match Percentage (2 pts)

Let's start by writing a function that will compute the percentage of matches found between a pair of given sequences. Note, you **cannot** assume that the sequences are of equal length. For example, given the strings

```
AGCTCCT
ACCTGT
```

the match percent would be 0.50 because 3 out of the 6 characters in the shorter string match the longer string. When computing the percentage, divide by the length of the shorter string. Round your result to 2 decimal places.

The function skeleton will look like this:

```
def matchPct(seq1, seq2):
   # Fill in the code to compute the percentage match
   # between the sequences
   return pct
```

Save the function in a file called `lab9_1.py` and submit it to Gradescope in the assignment called Lab 9-1. Your output should look like this:

```
>>> matchPct('AGCTCCT','ACCTGT')
0.5
>>> matchPct('AGGTC','AGGTC')
1.0
>>> matchPct('AAAA','BBBBB')
0.0
>>> matchPct('AAGTC','AGTC')
0.25
>>>
```

Instructor Initials: _____

II. Working with Substrings (2 pts)

1) In order to implement the sliding windows, we will have to access substrings from a Python string. Python has a way to do this. If you want to take a substring from position 2 to up to but not including position 5 in a string, you can use:

   `myString[2:5]`

   Try this out. In the Python shell, enter the following:

   `myString = "UNIVERSITY"`

   Now enter:

   `myString[2:5]`

   What does it return?

   "IVE"

   What **command** would you use if you wanted to return the match percentage between the first five characters of two given strings, `str1` and `str2`? (*Hint:* Call the function that you created in Part I above)

   matchPct(str1[0:5],str2[0:5])

Try this on these two strings:

```
str1 = "AACTCGTGAGTCT"
str2 = "ACTTGCGGGCTA"
```

What is the result of this command for these strings?

0.4

Instructor Initials: _____

III. Compute the Dot Plot (2 pts)

Let's now practice the technique to compute a dot plot with sliding windows. We will use a small example to demonstrate how this will work. If we assume a window size of **3** and a threshold of **0.5**, then we put a dot in the dot plot in position (0,0) if we have better than a 50% match between the first three characters in the first sequence and the first three characters in the second sequence. In this example, we would be comparing:

```
AGG
AAG
```

Since two of the three characters match, making it a 66% match, we would have a dot in cell (0,0) because the match is greater than 50%.

		0	1	2	3	4	5	6
		A	**G**	**G**	**T**	**A**	**A**	**G**
0	**A**	*				*		
1	**A**							
2	**G**							
3	**T**							
4	**A**							

Next we consider cell (0,1) in the dot plot. We slide the window on the top sequence over by one so are comparing:

```
GGT
AAG
```

In this comparison, we have no matches, so there is no dot placed in the cell (0,1).

Fill in the rest of the first row of the dot plot above.

When filling in cells (0,5) and (0,6), what problem do you encounter?

We are only comparing 3 characters at a time at the start of the second number (in this case 5 and 6). When looking for the two other characters after 5 and 6, there is nothing to compare, the index is out of range.

How can you address this problem?

When comparing, check that the (second number + 2) does not exceed past the string length. If it does, then you can't compare and write dots / asterisks.

Instructor Initials: _____

IV. Complete the Program (4 pts)

You can find skeleton code for the dot plot with sliding windows program on the Sample Code page for Week 11 (Thu).

Add your code for the `matchPct` function to the skeleton code, and then fill in the `computeDotPlot` function to implement the technique described above.

Save the program in a file called `lab9_2.py` and submit it to Gradescope in the assignment called Lab 9-2. Your output should look like this:

```
>>> main()
Enter top string: AGGTAAG
Enter side string: AAGTA
Enter size of window: 3
Enter expected threshold: 0.5
   A  G  G  T  A  A  G
A  *  -  -  -  *  -  -
A  *  *  -  -  -  *  -
G  -  -  *  -  -  -  *
T  -  -  -  *  -  -  -
A  *  -  -  -  *  *  -
```

Test the program using the following sequences with several different window sizes and different threshold values:

```
AAGGTAGCCTAACGTCCACTTTACCC
AGTAAGGTACCTACCTCAACTTCA
```

What do you observe about how the plot changes when you make the window larger?

What do you observe about how the plot changes when you make the threshold higher?

Instructor Initials: _____

CHALLENGE PROBLEMS

Every lab assignment, except for Lab 1, has at least one Challenge Problem. These exercises are meant to be a bit more challenging than the required work for the lab activity. These problems are meant to be done after you and your partner finish the required portion of the lab. By the end of the semester, every student is required to complete at least 4 Challenge Problems. This will count as a full lab grade. Any Challenge Problems completed beyond the required 4 will count as extra credit.

Assume you have two gene sequences of length *m* and *n* respectively. What is the general formula for the number of total alignments if we allow gaps in the shorter of the two sequences?

How much work is done to compute the optimal alignment score for these two sequences?

Instructor Initials: _____