CS380 Algorithm Design & Analysis
Assignment 5: Dynamic Programming

Date Assigned: Wednesday, April 8, 2015
Date Due: Monday, April 20 @ 9:15am
Total Points: 50 pts

For this assignment you will be implementing dynamic programming algorithms to provide answers to the questions listed below. The algorithms that you are to implement are:

- Longest Common Subsequence
- Sequence Alignment (either Needleman-Wunsch or Hirshberg)
- Horspool Search
- Perhaps others

"DNA.txt" is a file containing a number of DNA strands, each of length not more than 1024 characters. The first line in the file contains the number of DNA strands, followed by each of those strands separated by a new line.

"DNA_FRAGMENTS" is a file containing a number of DNA strands, each of length not more than 1024 characters, and DNA fragments, each of length not more than 128 characters. The first line in the file contains the number of DNA strands, followed by each of those strands separated by a new line. Next is the number of DNA fragments, followed by each of those fragments separated by a new line.

Requirements:

1. Display the longest common subsequence between each pair of DNA strands in the file "DNA.txt". Note that you will have to pair each strand with every other strand.
    a. **Input:** The file "DNA.txt".
    b. **Output:** The file "output_1.txt". The format of the file should be:

```
Strand X: xxxxxxx
Strand Y: xxxxxxx
LCS: xxxx
Length of LCS: xx
<blank line>
Strand X: xxxxxxx
Strand Y: xxxxxxx
LCS: xxxx
Length of LCS: xx
<blank line and continue in the same format>
```

2. Display the DNA strand pair with the longest common subsequence.
    a. **Input:** Your choice of either the file "DNA.txt" or the file "output_1.txt".
    b. **Output:** The file "output_2.txt". The format of the file should be:

```
The strands with the longest common subsequence are:
Strand A: xxxxxxx
Strand B: xxxxxxx
LCS: xxxx
Length of LCS: xx
<blank line> <output next pair if they are the same LCS>
```

3. Display the DNA strand pair with the longest common subsequence normalized against the maximum possible longest common subsequence for the pair. We are normalizing so that we can represent each value on the same scale. Example: if you find a subsequence of length 8 in a pair of strings of length 12 and 10 respectively, then the normalized length would be 8/10 (0.8), since the maximum possible subsequence length is 10 (the length of the shorter string).
    a. **Input:** Your choice of either the file "DNA.txt" or the file "output_1.txt".
    b. **Output:** The file "output_3.txt". The format of the file should be:
    ```
    The strands with the longest normalized common subsequence
    are:
    Strand A: xxxxxxx
    Strand B: xxxxxxx
    LCS: xxxx
    Normalized Length of LCS: xx
    <blank line> <output next pair if they are the same LCS>
    ```
4. Display the single longest common subsequence that is present in all of the DNA strands.
    a. **Input:** Your choice of either the file "DNA.txt" or the file "output_1.txt".
    b. **Output:** The file "output_4.txt". The format of the file should be:
    ```
    The single common longest common subsequence is:
    LCS: xxxx
    Length of LCS: xx
    <blank line> <output next LCS if it's the same length>
    ```
5. Display the sequence alignment for each pair of DNA strands. Use a gap penalty (δ) = 16 and the substitution matrix below.
    a. **Input:** The file "DNA.txt".
    b. **Output:** The file "output_5.txt". The format of the file should be:
    ```
    Strand A: ATTCA-CC-G-T-A
    Strand B: -TTCAA--TGGTC-
    Alignment Cost: xxxx
    <blank line> <continue outputting all pairs>
    ```

α	A	G	C	T
A	0	10	1	4
G		0	4	1
C			0	10
T				0

6. Display the number of editing operations needed to transform each DNA strand to each other one.
    a. **Input:** The file "DNA.txt".
    b. **Output:** The file "output_6.txt". The format of the file should be:
    ```
    Strand A: AT-C-TGAT
    Strand B: -TGCAT-A-
    Edit Distance: 5
    <blank line> <continue outputting all pairs>
    ```
7. Display the DNA strands with the smallest edit distance.
    a. **Input:** Your choice of either the file "DNA.txt" or the file "output_6.txt".
    b. **Output:** The file "output_7.txt". The format of the file should be:
    ```
    The pair of DNA strands with the smallest edit distance:
    Strand A: xxxxxx
    Strand B: xxxxxx
    Smallest Edit Distance: xxxx
    <blank line> <output next pair if they are also smallest>
    ```
8. Display the number of times that each DNA fragment appears in each DNA strand.
    a. **Input:** The file "DNA_FRAGMENTS.txt".
    b. **Output:** The file "output_8.txt". The format of the file should be:
    ```
    Strand: xxxxxx
    Fragment A: xx: #
    Fragment B: xx: #
    ```

```
Fragment C: xx: #
<blank line> <continue for all strands>
```

You are to implement a different driver for each of the above requirements.

- Each driver must be named (requirement_#) where # represents the requirement number.
- Write a brief paragraph for each question explaining which algorithms you chose to use and how you used the results of those algorithms. The paragraph must be in a comment at the top of the driver and be well written in complete English sentences.
- Output your results both to the screen and to a text file named "output_#", where # represents the requirement number. Do not print out the output!

What to Submit

I will pull your project out of Subversion. You must provide me with a color, double sided hard copy of all of your code. You must provide useful commit messages.

Record the number of hours spent on the assignment and place it in a text file in your project. Do not print it.

Your code must build without any warnings. You must follow the C++ coding standards. Check for memory leaks!