CS380 Algorithm Design & Analysis
Assignment 6: Dynamic Programming

Date Assigned: November 18, 20, 22, 25, in class.

Final Due Date: **DEC 3, 5pm!** Total Points: 70 pts

Group 1: Troy, Kai	Group 4: Andrew, Josh H.
Group 2: Liam, Chris	Group 5: Ashleigh, Alec
Group 3: Josh S., Thomas	Group 6: Ahmed, Nick, Ryan

This is your chance to put together the algorithms and object oriented programming techniques we have been studying this semester. It is up to your group to decide which algorithms to apply and how to interpret the results of those algorithms to answer each question. At a *minimum*, you will need to implement [1]Longest Common Subsequence, [2]Sequence Alignment (Needleman-Wunsch or Hirschberg), and [3]Horspool Search. For a bonus implement Knuth Morris Pratt Search (see your text book for this last one). You are free to implement any necessary helper functions or other algorithms. You will download a starter project from Subversion and work with a partner on this project in the CS Lab. **Do not download any code from the web for this assignment.**

You also need to write code to display a sequence alignment in the form:

```
-ca-ges
orange-
```

► At the end of each class, send Chadd an email detailing your current progress. This is a graded portion of the assignment.

In the starter project, you are provided with a number of files containing DNA strands, each of length not more than 1024 characters, and DNA fragments, each of length not more than 128 characters. Look at the text file to determine the data format. The file DNA.txt should be used for questions 1-8. The file Evolution.txt should be used for question 9. There are also a few test files with smaller DNA sequences and a driver in the Visual Studio project that allows you to build a new DNA file easily.

[1]Write a separate driver for each question below (named question_#.cpp). [2]Write a brief paragraph for each question explaining which algorithms you choose to use and how you used the results of those algorithms to answer the question. This paragraph must be in a comment at the top of the driver and be well written in complete English sentences. [3]Also include the answer to the question unless the question is "run this operation n*m times and show the result". *Save some paper.* I want to see answers such as "DNA strand 1 and DNA strand 2." Don't copy and paste the output of your driver into the comment. Output your results to the screen and to the file output_#.txt where # is the number of the question below you are solving. You may write as many drivers as necessary. You'll be sharing code through C++ classes.

For DNA comparisons, use the given cost matrix (which is completely fake) to evaluate differences in DNA strands. Use a *gap penalty* of 16.

Cost Matrix

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

1. What is the longest common subsequence between each pair of DNA strands? Display each pair with the subsequence exactly once.

2. Find the DNA strand pair with the longest common subsequence. Display the DNA pair and the subsequence.

3. Find the DNA strand pair with the longest common subsequence *normalized* against the maximum possible longest common subsequence for the pair. Display the DNA pair and the subsequence. Normalized in this context means representing each value on the same scale. For instance, 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 would be 10 (the length of the shorter string). You may choose to normalize your values in anyway you see fit.

4. Find and display the alignment for each pair of DNA strands using the given cost matrix.

5. What is the single longest common subsequence present in all DNA strands?

6. How many times does each DNA fragment appear in each DNA strand?

7. Which pair of the DNA strands are most similar? Provide the three most similar pairs (a particular DNA strand could appear in more than one pair). Determine "most similar" in any way you like and provide a justification for your choice.

8. The longest common subsequence may appear in more than one way in a pair of strings. For instance, the longest common subsequence for the following two strings is ABC (Strings: AAABBCCC, XAAXBXXC). There are a number of ways you can arrive at this same subsequence (using any of the first three letters in the first string, for example). Once you find the longest common subsequence, can you find the most compact way to produce that string? That is, find the subsequence in each string with the least distance between the first and last character in each string? Do this for each pair of DNA strands

9. **BONUS** Evolutionary biologists want to determine the order in which organisms evolved to build a *phylogenetic tree* or hierarchy. They often do this by ordering organisms by differences in DNA strands (fewer differences likely means more closely related). As an embarrassingly gross over-simplification of the problem, rather than build a tree, build a straight line between pairs of DNA strands to model the evolution from one strand to the other. Use the DNA in the file Evolution.txt for this problem. Use the given cost matrix and the sequence alignment cost.

   For each pair of DNA strands, can you determine a linear ordering among the given DNA strands that minimized the sum of the differences between successive strands? An example might be:
   AAAA and CAAA have an alignment cost of 1. CAAA and GCAA have an alignment cost of 6, for a total cost of 7. AAAA and GCAA have an alignment cost of 11, making the AAAA, CAAA, GCAA transition cheaper.

DOUBLE BONUS: Build the tree. You can use proper Bioinformatics techniques (Neighbor-Joining) or re-purpose an algorithm we have already used in class that will build the tree in a not terrible way.

What to Submit DEC 3, 5pm!

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 in SVN.

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

The write-up in the comments must be well written in complete English sentences. **Even if your group does not solve questions 8 and 9, I want a write up on how you would approach the problem (put this write up in drivers that do nothing).** You do not need to do write up anything for the Double Bonus if you don't solve the Double Bonus.

Get the files at: **svn+ssh://punetid@svnzeus/home/CS380/Group#_SVNROOT_DNA/CS380_DNA**

The bonus questions, and possibly other questions, might draw on algorithms outside of the string matching/similarity section.

You have four class sessions, I expect your group to take the time to build a well designed project.