CSCI 4830, Homework 1
Sequence alignment project
Due Tuesday, September 9, at 11:55pm, by Moodle

Begin, please, by reading over the slides from Wednesday on longest common subsequence (the second half of the slides cover this material); I'll resume those next week.

How similar are 2 strings of text? It's easy to say that BOBO == BOBO, but harder to describe the similarity between strings like BOBO and BONOBO, even though you can tell they share 4 letters in the same order (in two different ways). How might we quantify this similarity? And could we extend that quantification to longer text strings, which would be hard for us to match up by hand?

This method is sometimes called the "longest common subsequence" or LCS problem. It relies on the inductive observation that the best match to BOBO and BONOBO must depend on finding the best match at the beginning of the 2 strings (for instance, that first "B" should match) and continuing this match through to the best match for the rest of the 2 strings after that (OBO and ONOBO).

This handy insight points to a way to solve the problem by making a 2-dimensional array, with one string along the top and one along the side. This array has as many rows as the length of the first string plus one more row, and as many columns as the length of the second string plus one more row. We initialize it by making the first row and column all 0:

		B	O	N	O	B	O
	0	0	0	0	0	0	0
B	0						
O	0						
B	0						
O	0						

Now we fill in the table according to this rule: If there's a cell in the array at row i and column j, and the letters of string 1 at i-1 and string 2 at j-1 match, we take the maximum of three numbers: the number in row i - 1, column j; the number in row i, column j - 1; and 1 + the number in row i - 1, column j - 1. If there's no match, we take the maximum of the number in row i - 1, column j and the number in row i, column j – 1.

Filling in our example we get the following table, with the top score in the bottom right corner:

		B	O	N	O	B	O
	0	0	0	0	0	0	0
B	0	1	1	1	1	1	1
O	0	1	2	2	2	2	2
B	0	1	2	2	2	3	3
O	0	1	2	2	3	3	4

The 2 matches are:

```
BO - - BO          and          B - - OBO
| |    | |                      |     | | |
BONOBO                          BONOBO
```

To recover these instead of just getting the number of matching letters right, you'll need to do more bookkeeping.

Your code will take 2 strings from the user, preferably at the command line, and then match them to find the longest common subsequence. Print out the length of the longest common subsequence. Display the grid for this matching as well, but write that to a file.

Your code will also follow the description from the slides posted for lecture 2 (which posted early due to the projector failure) to backtrace the match and print the alignments with matches and gaps. This requires your code to remember how you obtained each match (whether you extended one or more of the diagonal, left, or upper matchings) and print pairs of alignments for the 2 sequences, like the examples here:

```
BO - - BO          and          B - - OBO
| |    | |                      |     | | |
BONOBO                          BONOBO
```

To get full credit for this assignment, you need to recover ALL of the possible longest matches. (I'd recommend a recursive solution here, at least to frame the problem). When you finish, upload your source code to the moodle. We'll grade this by testing and a very short interview after you hand it in.