11-756/18799D ASR: Assignment 2, DTW (Due 2/18/2015)

Problem 2(a)

Write a routine to compute the Levenshtein distance between two symbol strings.

The routine must optionally include two different pruning strategies (so that either may be turned on or off), one based on a maximum string edit distance of 3, and another based on a "beam" of 3 relative to the current best score in any column of the trellis.

Display the trellis, the scores and the best path in table format. An example table (without pruning) is shown below.

	*	s	i	t	t	i	n	g
*	0+	1	2	3	4	5	6	7
k	1	1+	2	3	4	5	6	7
i	2	2	1+	2	3	4	5	6
t	3	3	2	1+	2	3	4	5
t	4	4	3	2	1+	2	3	4
e	5	5	4	3	2	2+	3	4
n	6	6	5	4	3	3	2+	3+

Problem 2(b)

Extend the routine to simultaneously compare a given text string to multiple templates and select the best one. As an example, you may want to compare the text string "Eleaphent" to "Elephant", "Elegant" and "Sycophant" at the same time to determine which is the closest of the three. Do NOT use lexical trees -- templates are not to be merged to share common portions.

Apply the same pruning strategies as before (absolute distance of 3, and relative beam of 3). For the latter case, the best score and the pruning threshold must be computed across all templates.

Display the trellis and best path in table form. An example table for the word “Eleaphent” using pruning with relative beam of 3 is shown below, where -1 indicates locations that were pruned out. You will also need to show the tables for the other two words “Elegant” and “Sycophant”.

	*	E	l	e	a	p	h	e	n	t
*	0+	1	2	3	4	-1	-1	-1	-1	-1
E	1	0+	1	2	3	4	-1	-1	-1	-1
l	2	1	0+	1	2	3	4	5	-1	-1
e	3	2	1	0+	1+	2	3	4	5	-1
p	-1	3	2	1	1	1+	2	3	4	5
h	-1	-1	3	2	2	2	1+	2	3	4
a	-1	-1	-1	3	2	3	2	2+	3	4
n	-1	-1	-1	-1	3	3	3	3	2+	3
t	-1	-1	-1	-1	4	4	4	4	3	2+

Problem 2(c)

Using the code for the above problem, develop a spelling checker. For the spelling checker, you are provided a dictionary ("dict.txt") which is simply a list of word spellings. Each of the word spellings in your dictionary is now a template. Given some incoming text, each word is compared to the entire list of templates to determine the closest one. This is returned as the "spellchecked" version of the word.

Spell check this little Thai story: "typos.txt". Please run your code on the two provided dictionaries: "dict_5k.txt" and "dict_80k.txt".

Problem 2(d)

Adapt your levenshtein distance computation routine to compare word strings. Here, instead of comparing two characters at any node, you will compare two words (strcmp()). Also, include a backtrace in your code, so that you can find the exact alignment between the two word strings and count and report the total number of insertions, deletions and substitutions in the best alignment.

Use the above routine to compute the total errors in the original story above ("typos.txt") and the correct version of the story ("correct.txt"). Report the insertion, deletion and substitution errors.

Use the same routine to compute the total number of errors between your two spell-corrected versions of the story and the correct story.