CprE 315x

Programming Assignment #2

Due: 3/23/2012
Total 100 points.

In this assignment, you will develop a spell checking program. A word will be entered into your program as input. It will be compared with the list of words in the file "linux.words" using the sequence alignment algorithm. The closest (i.e., lowest cost) word and some information about the alignment will be reported. If there are several words with the same cost in the list, the word reported should be the earliest one in the list.

The penalties are defined as follows. For any two identical characters, the mismatch penalty is 0 (e.g., $\alpha_{AA} = 0$, $\alpha_{\#\#} = 0$). For two letters in the English alphabet that are the same but of different cases, the mismatch penalty is 1 (e.g., $\alpha_{aA} = 1$). For two different letters in the English alphabet with same case, the mismatch penalty is 2 (e.g., $\alpha_{ab} = 2$, $\alpha_{AB} = 2$). For two different letters in the English alphabet with different case, the mismatch penalty is 3 (e.g., $\alpha_{aB} = 3$, $\alpha_{Ab} = 3$). Otherwise, the mismatch penalty is 4 (e.g., $\alpha_{A\#} = 4$, $\alpha_{\#\$} = 4$). The gap penalty δ is 3.

Your program should be implemented in either C or C++. Please try your best to make your program fast and memory efficient. The input and output formats of your program (called "spell-check") are shown in the examples below. For submission, please email to the TA your source code and the outputs of your program for the words "Google", "ya-Hoo!", and "Microsoft".

Example 1: ocurrance

```
linux-3:/home/cnchu>spell-check ocurrance
Best match: currance
Cost: 3
Alignment:
-_o (3)
c_c (0)
u_u (0)
r_r (0)
r_r (0)
a_a (0)
n_n (0)
c_c (0)
e_e (0)
Dynamic Programming Sub-problem Cost:
          1   2   3   4   5   6   7   8   9
          o   c   u   r   r   a   n   c   e
   1   c   2   3   6   9  12  15  18  21  24
   2   u   5   4   3   6   9  12  15  18  21
   3   r   8   7   6   3   6   9  12  15  18
   4   r  11  10   9   6   3   6   9  12  15
   5   a  14  13  12   9   6   3   6   9  12
   6   n  17  16  15  12   9   6   3   6   9
   7   c  20  17  18  15  12   9   6   3   6
   8   e  23  20  19  18  15  12   9   6   3
```

Example 2: occurr#nce.

```
linux-3:/home/cnchu>spell-check occurr#nce.
Best match: occurrence
Cost: 7
Alignment:
o_o (0)
c_c (0)
c_c (0)
u_u (0)
r_r (0)
r_r (0)
e_# (4)
n_n (0)
c_c (0)
e_e (0)
-_. (3)
Dynamic Programming Sub-problem Cost:
```

		1	2	3	4	5	6	7	8	9	10	11
		o	c	c	u	r	r	#	n	c	e	.
1	o	0	3	6	9	12	15	18	21	24	27	30
2	c	3	0	3	6	9	12	15	18	21	24	27
3	c	6	3	0	3	6	9	12	15	18	21	24
4	u	9	6	3	0	3	6	9	12	15	18	21
5	r	12	9	6	3	0	3	6	9	12	15	18
6	r	15	12	9	6	3	0	3	6	9	12	15
7	e	18	15	12	9	6	3	4	5	8	9	12
8	n	21	18	15	12	9	6	7	4	7	10	13
9	c	24	21	18	15	12	9	10	7	4	7	10
10	e	27	24	21	18	15	12	13	10	7	4	7

Example 3: Google

```
linux-3:/home/cnchu>spell-check Google
Best match: goggle
Cost: 3
Alignment:
g_G (1)
o_o (0)
g_o (2)
g_g (0)
l_l (0)
e_e (0)
Dynamic Programming Sub-problem Cost:
```

		1	2	3	4	5	6
		G	o	o	g	l	e
1	g	1	4	7	9	12	15
2	o	4	1	4	7	10	13
3	g	7	4	3	4	7	10
4	g	10	7	6	3	6	9
5	l	13	10	9	6	3	6
6	e	16	13	12	9	6	3