CSE250, Spring 2014 Assignment 3 Due Fri. 2/21

This assignment has **both** a hardcopy part and an online-submission part. The hardcopy part is due *in class* on **Friday 2/21**, while the online part it due at the (UB CSE-traditional?) **midnight time**.

Reading. For next week, follow the "Week 3 and beyond" part of the reading directions given in the longer posted version of Assignment 1. That is, read the supplementary notes and links on O, o, Θ, Ω notation now on the course webpage (the first is the same as the 1-sheet handout given Fri. 2/14), and then make the transtion from Section 2.6 of the text to the more-extensive treatment on my lecture slides, including L'Hôpital's Rule. Finally, really looking ahead to the end of the week and the following weekm please read the first two sections of Chapter 3.

(1) Text, exercise 1 on pages 178–179. Note that the text is saying "$O(n)$" when it really means $\Theta(n)$—thus you cannot get away with the trivial (though true!) answer that all four times are $O(n^2)$! $(2 + 2 + 2 + 3 = 9$ pts.)

(2) Suppose $T(n) = n^3 + 5n^2 + 20n - 10$. Note that this differs from the text in exercise 2 on page 179 in having $+5n^2$ not $-5n^2$. To show formally that $T(n) = O(n^3)$, we have to exhibit constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $T(n) \leq cn^3$.

First consider $c = 2$, and find the minimum value of n_0 that works. Then try $c = 1.1$. How big is the minimum value of n_0 now? $(6 + 6 = 12$ pts.)

(3) This and the next problem are hardcopy versions of routines that will added to programming parts on the next set. Write on paper a routine `hd1` that takes two `string` variables x and y and tests whether x differs from y in exactly one place. For example, x can be `rain` and y can be `pain` or `rein` or `rail`. Write your routine in such a way that you could change it to return `true` also when x and y are the same, just by changing part of one line of code. (12+3 = 15 pts.)

(4) With reference to Problem (3), now code a routine

```
bool xd1(const string& x, const string& y)
```

that also allows for the case that x or y is a one-letter extension of the other. For instance, if x is `raid` then y can be `braid` or `raids` or `aid`. For the moment we are not allowing internal inserts like `rabid` or internal deletions like `rid`. If x and y have the same length, then the answer should be the same as for `hd1(x,y)`, while `xd1` should not "allow" (that is, return `true` in) the case that both an extension and a flip occur, as with $y =$ `brain`.

Write your code so that it calls `hd1` one or more times. Note that the relation expressed by `xd1` is symmetric—it should give the same answer if you switch x and y. Show, however, that you could save a line or so of code if you `REQ`uire that x never be longer than y in any call. Would this also save you a function call to `hd1`? Write a couple of sentences with your analysis and opinion of whether this last kind of tradeoff in logic/reliability for greater code efficiency is "worth it." $(15 + 9 = 24$ pts., for 60 total on the hardcopy part—**more is**

overleaf. FYI, `hd` stands for "Hamming distance" and `xd` for "extension distance"—we will also have `ed` for "edit distance.")

Programming Assignment—due online Fri. 2/21, 11:59pm

(5) Derive a new class `PeekDeque` from your `StringDeque` that adds the following functionality: it allows client-controlled reading of strings inside the deque without having to destructively pop to get to them. The functionality should be provided *without* making an `int` or `size_t` value of the new `peekIndex` variable visible to the end user. Instead your library class should add methods `moveFrontward()`, `moveRearward()`, and `peek()`, with the last returning the string value at the current `peekIndex` location.

Also override the two pop methods, just to show an example of overriding. They should change the error message so that it says "Attempt to pop from empty PeekDeque." OK not to do any change to the push methods—this is just for show. It is also OK to have the code all in one file `PeekDequeNNN.cpp`, where `NNN` are your initials—we will make separate `.h` and `.cpp` files for the class hierarchy later. Finally rewrite `main` to include peeking and printing just the words that begin with '`s`'. This part is 48 pts., for 108 total on the assignment. Again there is just one file that you need to submit, by

```
submit_cse250 PeekDequeNNN.cpp
```