# CS 2433 C/C++ Programming

Programming assignment 5

10 points

Due date and time: 11:59 PM, September 25, 2016

The objective of this assignment is to familiarize you with the use of pointers, as well as implementing and using functions. The program for this assignment will perform the same functions at the user interface as the program you submitted for program 4, but will do so in a different manner. Specifically, where the two lists used in program 4 were managed and traversed using indices into the static array of nodes, pointers will be used in managing and navigating those lists. This will require some revision to your program 4 code, but other revisions will be required in order to create and use function you will write to perform some of the tasks of managing the two lists.

Write a C program, in a file named 'p5.c', to solve the following:

Similar to the **wnode typedef struct** in program 4, declare a **typedef** for a **struct** that contains an array of 16 characters, and two **pointers**. The **typedef** is to be named **wnode**, the character array **aword** and the pointers **last** and **next**. The pointers must be of either type **wnode *** or **void ***. Declare an array of 100 nodes of **typedef wnode** named **listnodes**.

Your program will operate at the user level exactly as program 4 operated: insert printable words that do not start with a tilde, delete words immediately preceded by a tilde, and stop performing insertions and deletions when "!stop" is read from **stdin**.

Internally, you must use the pointer fields of the **wnodes** (**next** and **last**) in both lists as pointers, the low-level operations for insertion and deletion from these lists are the same, except for the fact that you will be using pointers. The **listnodes[0]** node is still the head-node for the doubly linked list, but **freenodes** must be a pointer to a node, the free-node list linked using the **next** pointer of each node, and the marker for an empty list and end of the free node list must be a null pointer (**(wnodes *) NULL**, for example).

Changes in the code will be required. You will implement the following functions and use them to manage the doubly-linked word list and the singly-linked free list.

| | |
|---|---|
| **getFreeNode** | Returns either a null pointer of type **wnode *** or a pointer to a node of that type that was freed from the free node list. Return of a null pointer indicates that there were no free nodes on the free list. |
| **setNodeFree** | Places a node (passed to the function via a pointer to the node) that is not in use in the doubly linked list onto the list of free nodes. This function is of type **void** (in other words is does not return anything). |
| **insertWord** | Inserts a node with a word in its **aword** field in its proper place in the doubly-linked list. The node to be inserted is passed to this function as a pointer to that node. This function is of type **void**. |

**deleteWord** This function either deletes a node containing the word (a **char** string) passed to this function (as a pointer to the **char** string) from the doubly linked list. This function returns either a pointer to the node thus freed or a null pointer, if the word was not found.

You must determine any other parameters that are to be passed to these functions beyond those explicitly called for above.

REQUIREMENTS:

1) Program must compile without errors and start without crashing. (10 points.)
2) Program source must be properly documented, with the standard student and assignment information at the start of the file. Documentation includes a block of comments immediately before each function describing the function inputs and outputs, plus "in-line" documentation of the code. (2 points.)
3) You may assume the list will not be empty at normal termination of processing, but you must print the in-order and reverse-order lists correctly as per the instructions for program 4. (8 points.)

CAUTION: This is an individual programming assignment. All work must be done individually. Sharing (or copying from any source) of code will be treated as plagiarism and dealt with accordingly.

Submit the solutions using the "handin" command.

Example submission command is "handin cs2433 program5 p5.c", where p5.c is the file being submitted.