CSCI 340 **Computer Assignment 4** **Fall 2017**
(10 points)

Word Scanner

For this computer assignment, you are to write and implement an interactive C++ program to scan and process a stream of words of a plain text.

Your program should start with an empty list of words. (Combinations of blanks, tabs, and newline characters to separate words.) After reading each word from the input stream, do the followings:

Check the word for punctuation marks. If the first letter of the word is preceded or its last letter is followed by punctuation marks, delete them from the word; however, if a word contains punctuation marks in the middle, ignore the letters beyond these punctuation marks. For example, if the word is fish, then the resulting word should be fish; however, if the word is fish_net then the resulting word should still be fish.

Programming Notes:

1. Use a map container to store the words, where each word is represented by a pair < string, int >. The first element of the pair, named first, contains the name of the input word, and the second element of the pair, named second, contains its frequency in the input stream. To use a map container in your program, you need to insert the statement: #include <map> in your header file, and to use the functions in the STL, you also need to insert the statement: #include <algorithm> in your header file.

2. In addition to the main ( ) routine, implement the following subroutines in your program:

   - void get_words (map < string, int >&) : It gets a word from the input stream and removes its punctuation marks.

   - void print_words (const map < string, int >&) : It prints the final list of words and their frequencies. It also prints the number of nonempty words and the number of distinct words in the input stream.

   - void clean_entry (const string&, string&) : It cleans a word from its punctuation marks. The first argument is the original word in the input stream and the second argument contains the same word after cleaning.

3. The main ( ) routine calls the subroutine get_words ( ) to get words from the input stream, and to clean an individual word from its punctuation marks, get_words ( ) calls the subroutine clean_entry ( ).

4. You can check the size of a string by the member function length ( ). For string s, the statement: s.length ( ) returns the total number of characters in s. If the returned length is 0 (i.e., the original word contains nothing but punctuation marks), simply ignore the word and read in the next one.

5. Use a map container to store the cleaned words (but not the empty words). The statement: m [s]++ copies the string s into the map m and updates its frequency. Remember a map keeps only one copy of each individual item (in sorted order) specified in its first argument, key, and stores its frequency in its second argument, value. When key is used as an index to a map, it returns the corresponding value in the map.

6. You can implement the subroutine clean_entry ( ) as follows:

    a. Initialize an index variable to indicate the first character in a word, and then scan the word until an alphanumeric character is found. You can use the C library function isalnum ( ) to check a character in the input string. If the character is alphanumeric, this function returns true; otherwise, it returns false. Initialize another index variable for this position and continue scanning the word until the first punctuation mark. If the word does not contain any alphanumeric characters, you must stop scanning the word when you reach the end of the word. An alternative approach to find the values of these two index variables is to use the function find_if ( ) in the STL. In this case, you need to have two predicates: one of them returns true if a character is alphanumeric but the other one returns true if the character is non-alphanumeric, and each of these predicates can be used as the third argument to find_if ( ) function.

    b. At this point you have two index variables: the first one indicates the position of the first alphanumeric character and the second one indicates the position just after the last alphanumeric character. If the word does not have any alphanumeric characters, then both index values indicate the same position.

    c. Copy the word from the first alphanumeric character until the last one to another string. Use the member function substr ( ) for this task. For string s, the statement: s.substr (pos, len) returns the substring from s, where the index value pos indicates the starting position of the substring and len specifies its size.

    d. Finally, convert all capital letters in the cleaned word to lowercase letters. You can convert a character to lowercase by the C library function tolower ( ). Instead of using a loop for this task, try to use the function for_each ( ) in the STL, where you need to have a unary function as the third argument to for_each ( ) function.

7. Print the number of nonempty words in the input stream and the number of distinct words. You can obtain the number of elements in a map by the member function size ( ). Also print the contents of your map: name of each word and its frequency (NO_ITEMS words per line and the words are left aligned). To allocate certain number of spaces for an output value, use the function setw (ITEM_W), and for its left alignment, use left in your cout statement. In your program, use NO_ITEMS = 3, and ITEM_W = 16. For printing, the main ( ) routine calls the subroutine print_words ( ).

8. To compile and link your program with the system library routines, execute: Make N=4. To test your program, execute: Make execute N=4. This command executes your program with the input file prog4.d in directory: ~cs340/progs/17f/p4, and displays

the output as well as any error messages both on the terminal screen and in prog4.out. After you are done with the program, you don't need its object and executable files any more. To delete them, execute: Make clean.

9. You can find the correct output of this program in the output file prog4.out in directory: ~cs340/progs/17f/p4.

10. When your program is ready, send its source and header files to your TA, by executing: mail_prog.340 prog4.cc prog4.h.