# CS 2433 C/C++ Programming

Programming assignment 6
20 points
Due date and time: 11:59 PM, October 18, 2016

The objective of this assignment is to further familiarize with functions, pointers and parameter passing, as well as to familiarize you with file I/O.

For this assignment, you are required to solve the problem as defined below:

PROBLEM: Read a file containing text replacements, file name given as the first command line argument upon program invocation. Each line will contain one or two strings (such strings will not contain white-space characters), the first identifying a string (or pattern) to be searched for, and the second (if present) the string to replace the pattern with. If no second string is present in the line, the pattern is to simply be removed. You may assume that there will be no more than FIVE lines in this first input file, and the two strings in each line no longer than 15 chars in length, each. Read lines of text from a second input file specified as the second command line argument when the program is invoked. Remove each of the patterns read from the first input file, and when a replacement string is provided insert the replacement string. Write the modified strings to the output file. The name of the output file is the third command line argument when the program is invoked. Remove the patterns, and insert any specified replacements, in the order the patterns were read from the first input file. The process is to read patterns and optional replacement strings from the first input file and save these in an array of char arrays, then read the second input file one line at a time, perform all required replacements or removals in the order they were listed in the first input file, and write the revised lines to the output file, one line at a time.

The function **rplpattern** must be passed three parameters. The first is a pointer to the string to be searched for a pattern. The second is a pointer to the pattern to be searched for. The third parameter is a pointer to a string to be used to replace the pattern. The third parameter may be a null pointer, in which case no replacement is to be performed, a pointer to an empty string, again indicating no replacement is to be performed, or a pointer to a string up to 15 chars in length, which is to be used to replace the pattern in the string searched. Note that the replacement string may be longer than the pattern. This means you must allocate a char array within the **rplpattern** function to act as a buffer into which you copy chars as required to construct the modified string. When the search and replacement operations are completed, the resulting string is to be copied to the location indicated by the first parameter, which pointed to the input string.

You may assume that there are no overlaps of similar patterns. You may also assume that the string patterns do not cross line boundaries and the maximum line length is 256 characters. You are required to open, read and appropriately store the up to five lines from the first input file, then open the second input file, read the contents one line at a time, remove/replace the patterns if they are present (possible to have multiple instances

per line), and write the resulting line to the output file. The two input and one output file names should be passed to the main program as command line arguments, as described above. Removal or replacement of occurrences of a pattern from a string must be done using a function named **rplpattern**. You are required to implement this function. The function takes a string and a pattern as parameters and removes or replaces all occurrences of a pattern in a string and returns the resulting string as a parameter. The source string, the pattern, and replacement strings must be parameters (arguments). The function returns 0 if the pattern is not found in the source string, otherwise it returns the number of times the pattern was found. Upon return from **rplpattern**, the number of occurrences of the pattern is written to the **stdout** stream, without additional text or numbering, two space characters between each number. (DO NOT insert a new-line character ('\n') between the numbers returned for an individual line's three passes through **rplpattern**, but insert a new-line character after the last call to **rplpattern** returns the number of removals/replacments.

The main program opens the files, reads patterns and replacements, reads lines from the input file, calls the function **rplpattern** to remove or replace patterns one at a time, and writes the resulting string in the output file, reporting the number of patterns removed or replaced during the process.

REQUIREMENTS:

1) Programs that do not compile, or that crash with exceptions, receive no credit.
2) Implement a function **rplpattern** as specified above that takes a string, a pattern and an options replacement string, then creates a string free of the pattern, returns 0 if pattern is not found, or the number of occurrences otherwise. The source, pattern, and replacement are formal parameters of type char array (7 points).
3) Write the revised text to the output file with correct replacement or removal of patterns (7 points).
4) The counts of patterns removed or replaced from each line must be printed to the **stdout** stream, up to five per line of input file (penalty 1 point).
5) File names must be command line arguments (penalty 1 point).
6) If either input file is empty, print an appropriate message and exit, without creating an output file (penalty 1 point).
7) Remove all occurrences of the patterns in the specified order using the function **rplpattern** (2 points).
8) Document program segments appropriately (penalty 1 point). Identify the segments – open files, read a line, remove or replace patterns, write line free of patterns, etc.

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

Submit the solutions using the "handin" command.

Example submission command is “handin cs2433 program6 prog6.c”. prog6.c is the file being submitted.