CptS 121 - Program Design and Development

Bonus Assignment: String Processor - worth up to 2% of your final grade

Due: Friday, June 14th, 2019 by midnight; send as attachment to beiyu.lin@wsu.edu

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:
- Declare and define pointers to strings
- Manipulate strings

II. Prerequisites:

Before starting this programming assignment, participants should be able to:
- Apply and implement pointers in C
- Pass output parameters to functions
- Analyze a basic set of requirements and apply top-down design principles for a problem
- Apply repetition structures within an algorithm
- Construct while (), for (), or do-while () loops in C
- Compose C programs consisting of sequential, conditional, and iterative statements
- Eliminate redundancy within a program by applying loops and functions
- Create structure charts for a given problem
- Open and close files
- Read, write to, and update files
- Manipulate file handles
- Apply standard library functions: fopen (), fclose (), fscanf (), and fprintf ()
- Apply and implement pointers 2-dimenional arrays
- Define and apply structs in C
- Compose decision statements ("if" conditional statements)
- Create and utilize compound conditions
- Summarize topics from Hanly & Koffman Chapter 8 including:
  - What is an array?
  - Distinguishing between single dimensional and 2-dimentional arrays
  - What is an index?

III. Overview & Requirements:

Write a program that performs string processing on all strings. NOTE: You may **NOT** use the standard library functions found in <string.h>. Also, you must use only *pointer* notation and pointer *arithmetic*, where appropriate! You may **NOT** use array notation! Your program should define the following functions:

- (6 pts) char *my_fgets (char *s, int n, FILE *stream) — Reads at most one less than the number of characters specified by `n`, into the array pointed to by `s`, from the input pointed to by `stream`. No more characters are

read after a newline is encountered or end of file is encountered. The newline if encountered is stored by the array pointed to by `s`. A null character is appended to the end of string `s`. If the function is successful in reading characters into `s`, then `s` is returned; otherwise a NULL pointer is returned.

- (6 pts) int my_fputs (const char *s, FILE *stream) - Writes `s` to the output specified by `stream`. If the function is successful, then it returns the number of characters written to the output; otherwise it returns MY_EOF (i.e. for cases where `stream` does not point any where yet). Make sure you #define MY_EOF as -1.
- (6 pts) int my_fgetc (FILE *stream) - Reads the next character from the input pointed to by `stream`. If the input is at the end of file or a processing error occurs return MY_EOF; otherwise return the integer representation of the character read.
- (6 pts) int my_fputc (int c, FILE *stream) - Writes the character `c` (converted to a character) to the output specified by `stream`. If the write is successful the ascii value of the character is returned; otherwise MY_EOF is returned.
- (6 pts) char *my_gets (char *s) - Reads characters from `stdin` into the array pointed to by `s` until a newline is encountered. The newline character is NOT kept at the end of the array pointed to by `s`. A null character is written to the end of string `s`. The function returns `s`.
- (6 pts) int my_puts (const char *s) - Writes the string pointed to by `s` to `stdout`. The function appends a newline to the output. The function returns the number of characters written to the output.
- (6 pts) int my_getchar (void) - Returns the ascii value of the next character read from `stdin`.
- (6 pts) int my_putchar (int c) - The function writes character `c` to `stdout`. The character `c` is returned.
- (6 pts) char *my_strcpy (char *destination, const char *source) - Copies all characters in the array pointed to by `source` into the array pointed to by `destination`. The null character is also copied. The function returns `destination`.
- (6 pts) char *my_strncpy (char *destination, const char *source, int n) - Copies no more than `n` characters from the string pointed to by `source` into the array pointed to by `destination`. The function does not copy any characters past a null character. If the string pointed to by `source` is less than `n` characters, null characters are appended to the end of the array pointed to by `destination` until `n` characters have been written.
- (6 pts) char *my_strcat (char *destination, const char *source) - This function appends a copy of the string pointed to by `source` (including the null character) to the end of the string pointed to by `destination`. The append overwrites the null character at the end of `destination`. The string pointed to by `destination` is returned.
- (6 pts) char *my_strncat (char *destination, const char *source, int n) - This function appends no more than `n` characters from the string pointed to by `source` to the end of the array pointed to by `destination`. The null character is appended to the end of the result. The `destination` pointer is returned.
- (6 pts) int my_strcmp (const char *s1, const char *s2) - This function compares the string pointed to by `s1` to the string pointed to by `s2`. If the string pointed to by `s1` comes before the string pointed to by `s2` in dictionary ordering, then -1 is returned. If the string pointed to by `s1` is the same as the string pointed to by `s2`, then 0 is returned (the compare function is case sensitive). Otherwise 1 is returned.

- (6 pts) int my_strncmp (const char *s1, const char *s2, int n) - This function compares no more than `n` characters (characters following a null character are not compared) from the string pointed to by `s1` to the string pointed to by `s2`. If the string pointed to by `s1` comes before the string pointed to by `s2` in dictionary ordering, then -1 is returned. If the string pointed to by `s1` is the same as the string pointed to by `s2`, then 0 is returned (the compare function is case sensitive). Otherwise 1 is returned.
- (6 pts) int my_strlen (const char *s) - This function returns the length of the string pointed to by `s`. The computation of length does NOT include the null character.

- (10 pts) A main function that tests each of the above implemented functions. You decide how to test your program, making sure to develop good test cases. You will be graded in part on the quality of your test cases, that is, the extent to which they cover the possible range of inputs.

IV. Submitting Assignments:

1. Please send your solution as a .zip attachment to beiyu.lin@wsu.edu.
2. Your project must contain one header file (a .h file), two C source files (which must be .c files), and project workspace. **Be sure to delete both debug folders before you send your solution via email.**
3. Your project must build properly. The most points an assignment can receive if it does not build properly is 65 out of 100.

V. Grading Guidelines:

This assignment is worth 100 points or 2% of your final grade. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

1. (90 pts) for adherence to the above requirements (please see the individual point totals above)
2. (10 pts) for "good" style and design (i.e. proper function declarations, definitions, and comments)