# Lab 07: "Manual"

Due: **07:59 AM** on **Friday, March 24**
Points: **100** points

**Objective**:

- To design and implement functions to process characters and strings.

**Instructions:**

- Be sure to document your code (add comments on top of each function).
- In the comments add your name, date, course, homework number, and statement of problem.
- Once you are done, upload your final solution through Blackboard.
- Implement a program called `Manual.c` that implements the following functions.
- Don't change name functions names, all functions names given should be kept as it is.

**Steps:**

- It is up to you to choose what the return type should be. Be creative!
- Write your own test drive for testing all functions.
- Feel free to use and reference strings functions explained in Ch.08.
- `Manual.c` has the following functions

1. ***(Displaying Strings in Uppercase and Lowercase)*** Write a function called `upperLower` that inputs a line of text into char array `s[100]`. Output the line in uppercase letters and in lowercase letters.

2. ***(Converting Strings to Integers for Calculations)*** Write a function called `convertStrtoInt` that inputs four strings that represent integers, converts the strings to integers, sums the values and prints the total of the four values.

3. ***(Converting Strings to Floating Point for Calculations)*** Write a function called `convertStrtoFloat` that inputs four strings that represent floating-point values, converts the strings to double values, sums the values and prints the total of the four values.

4. ***(Comparing Strings)*** Write a function called `compareStr` that uses function `strcmp` to compare two strings input by the user. The function should state whether the first string is less than, equal to or greater than the second string.

5. ***(Comparing Portions of Strings)*** Write a function called `comparePartialStr` that uses function `strncmp` to compare two strings input by the user. The function should input the number of characters to be compared, then display whether the first string is less than, equal to or greater than the second string.

6. ***(Random Sentences)*** Write a function called `randomize` that uses random number generation to create sentences. The function should use four arrays of pointers to char called article, noun, verb and preposition. The function should create a sentence by selecting a word at random from each array in the following order: article, noun, verb, preposition, article and noun. As each word is picked, it should be concatenated to the previous words in an array large enough to hold the entire sentence. The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with a period. The function should generate 20 such sentences. The arrays should be filled as follows: The article array should contain the articles "the", "a", "one", "some" and "any"; the noun array should contain the nouns "boy", "girl", "dog", "town" and "car"; the verb array should contain the verbs "drove", "jumped", "ran", "walked" and "skipped"; the preposition array should contain the prepositions "to", "from", "over", "under" and "on". After the preceding function is written and working, modify it to produce a short story consisting of several of these sentences.

7. ***(Tokenizing Telephone Numbers)*** Write a function called `tokenizeTelNum` that inputs a telephone number as a string in the form (555) 555-5555. The function should use function `strtok` to extract the area code as a token, the first three digits of the phone number as a token and the last four digits of the phone number as a token. The seven digits of the phone number should be concatenated into one string. The function should convert the area-code string to int and convert the phone-number string to long. Both the area code and the phone number should be printed.

8. ***(Displaying a Sentence with Its Words Reversed)*** Write a function called `reverse` that inputs a line of text, tokenizes the line with function `strtok` and outputs the tokens in reverse order.

9. ***(Counting the Occurrences of a Substring***) Write a function called `countSubstr` that inputs several lines of text and a search string and uses function `strstr` to determine the total occurrences of the string in the lines of text. Print the result.

10. ***(Counting the Occurrences of a Character)*** Write a function called `countChar` that inputs several lines of text and a search character and uses function `strchr` to determine the total occurrences of the character in the lines of text.

11. ***(Counting the Letters of the Alphabet in a String)*** Write a function called `countAlpha` based on `countChar` that inputs several lines of text and uses function `strchr` to determine the total occurrences of each letter of the alphabet in the lines of text. Uppercase and lowercase letters should be counted together. Store the totals for each letter in an array and print the values in tabular format after the totals have been determined.

12. ***(Counting the Number of Words in a String)*** Write a function called `countWords` that inputs several lines of text and uses `strtok` to count the total number of words. Assume that the words are separated by either spaces or newline characters.

13. ***(Strings Starting with "b")*** Write a function called `startsWithB` that reads a series of strings and prints only those beginning with the letter "b".

14. ***(Strings Ending with "ed")*** Write a function called `endsWithed` reads a series of strings and prints only those that end with the letters "ed".