# Assignment 05: *Exploring Functions, Strings, Arrays, and File I/O*.

**CS 270- Fall 2024**

| Project Available | | Nov. 18 |
|---|---|---|
| | **Points** | **Due Date(at 11:59 pm)** |
| **Project Total** | 20 | (Wednesday) Nov. 27 |

***Read the document carefully and thoroughly, before you start!***

**Objective:** In this assignment, you will develop your skills in C programming by working with functions, input/output from standard input and output, file handling, command-line options, strings, and arrays. Starter programs are provided to help you get started, including an example that demonstrates reading data from a file as a string. Carefully review and execute this example before modifying it to complete the tasks outlined in the assignment. Your work will be assessed based on its functional correctness, efficient use of resources, and adherence to good coding practices, including properly formatted source code and meaningful comments (header, function, and inline). Avoid redundant code by effectively using functions, and ensure your solution minimizes unnecessary function calls and memory usage. Your programs must compile and run error-free on the *compute.cs* system using the GCC compiler. I *highly recommend* focusing on clarity and efficiency in your implementation to maximize your grade.

**Submission Guideline**: For this assignment, you are required to submit your source code files with a ***.c*** extension on Canvas. Additionally, you must provide proof that your code compiles successfully. This proof can be submitted as screenshots showing the successful compilation of your programs. Ensure that all the screenshots are consolidated into a ***single PDF*** file before uploading. If you are unsure how to submit your assignment or encounter any issues during the process, please reach out to me for clarification or assistance well before the deadline.

**Useful C-library Functions that would be helpful for dealing with Assignment**: Below are some useful functions and documentation references to assist you in writing your program. For detailed information about these commands, consult the system's manual pages or search online. For example, you can type man *fgets* in the terminal to access the manual for the *fgets* function.

**Functions from the stdio.h Library**

- **printf()**: Print formatted output to the standard output (stdout). *(Manual page: man 3 printf)*
- **fprintf()**: Similar to printf(), but allows output to any file.
- **fopen()**: Open a file and return a file pointer.
- **fclose()**: Close an open file.

- **feof()**: Check for the end-of-file (EOF) condition on a file stream.
- **fgets()**: Read a line of input (e.g., from stdin or a file).
- **fflush()**: Flush the buffered output to the screen or a file. *(Example: fflush(NULL) flushes all output streams.)*

**Functions from the ctype.h Library**

- **isdigit()**: Check if a character is a digit (0-9).
- **isspace()**: Check if a character is whitespace (includes spaces, tabs, and newlines).
- **isupper()**: Check if a character is an uppercase letter (A-Z).
- **islower()**: Check if a character is a lowercase letter (a-z).
- **isblank()**: Check if a character is a space or tab.
- **isalpha()**: Check if a character is alphabetic (A-Z or a-z).
- **toupper()**: Convert a character to uppercase (a-z to A-Z).
- **tolower()**: Convert a character to lowercase (A-Z to a-z).

**Functions from the string.h Library**

- **strlen()**: Calculate the length of a string.
- **strdup()**: Create a duplicate of a string (returns a pointer to the copy).
- **strchr()**: Find the first occurrence of a character in a string.
- **strcmp()**: Compare two strings for equality.
- **strncmp()**: Compare a specific number of characters from two strings.

**Functions from the stdlib.h Library**

- **strtol()**: Convert a string to an integer.
- **strtod()**: Convert a string to a floating-point number.

*Note*: Avoid using deprecated functions like *atoi* and *atof*. Instead, use *strtol* and *strtod*, which are more robust and supported across systems.

These functions will help you handle input, output, string manipulation, and type conversion effectively in your C programs.

**Assignment Tasks Overview**

**1. Starter Program**

Before implementing the functionality to meet the project specifications, you should first ensure that you can successfully run the provided code for handling data from standard input (**stdin**).

Begin by downloading the project template zip file and practicing with the given programs located in the prog1/ and prog2/ directories.

The provided programs read data from **stdin**, process it by removing any newline characters (if present), and then print the cleaned lines back to the console. Additionally, the programs count and display the total number of lines processed and printed.

To streamline testing, instead of typing input manually into the console, you can write the input data to a file and redirect it as **stdin** when running the program. This method is a convenient and powerful way to test programs efficiently. Below is an example of how to redirect a file as input:

```
./your_program < test_input.txt
```

You should create several test files with varying content and use them to evaluate how the program handles different cases, including:

- Regular input lines.
- Empty lines.
- Very long lines (to check how the program handles them).

This approach will help you identify any potential issues and ensure the program behaves as expected under different scenarios. Testing with a variety of input files is crucial for verifying correctness and robustness.

**2. Program 01: palindrome.c**

Write a program that processes lines of input from the user and checks if each line is a palindrome. The program should handle advanced test cases by ignoring differences in case, punctuation, and whitespace. Blank lines should be skipped and should not be counted as palindromes or non-palindromes.

To receive full credit:

1. Implement a separate function that takes a string as input and returns:
    - 1 if the string is a palindrome.
    - 0 otherwise.
2. The main function should handle all input and output operations.
    - It should read lines from the standard input, call the palindrome-checking function, and display whether each line is a palindrome.
    - At the end, the main function should print a summary of how many lines were processed and how many were palindromes.

```
shell$ cat test-file0

Madam, in Eden, I'm Adam.

Hello, World!

A man, a plan, a canal, Panama!

Was it a car or a cat I saw?

Racecar

Step on no pets

Able was I, I saw Elba

Not a palindrome here

shell$


Output:

[Palindrome!]: Madam, in Eden, I'm Adam.

[Regular String]: Hello, World!

[Palindrome!]: A man, a plan, a canal, Panama!

[Palindrome!]: Was it a car or a cat I saw?

[Palindrome!]: Racecar

[Palindrome!]: Step on no pets

[Palindrome!]: Able was I, I saw Elba

[Regular String]: Not a palindrome here

Summary:

Total Palindromes: 6 (out of 8 strings)
```

**3. Program 2: Caesar Cipher (cipher.c): Encrypting and Decrypting a Text.**

Your task is to write a program that performs encryption and decryption using the **Caesar cipher** method. The program will take two command-line arguments to determine the operation (encryption or decryption) and the shift amount. Input will be provided through **stdin**, and the processed output will be displayed on **stdout**.

**Program Requirements**

a. **Command-Line Arguments:**

  - The first argument specifies the operation:
    - -e: Encrypt the input.
    - -d: Decrypt the input.
  - The second argument specifies the amount to shift the alphabet during encryption or decryption.

b. **Input and Output:**

  - **Input** will be provided via standard input (**stdin**).
  - The program should output the result to standard output (**stdout**).

c. **Case and Character Handling:**

  - Ignore the case of input characters (treat all input as uppercase).
  - Preserve whitespace and any non-alphabet characters (these should remain unchanged in the output).
  - Output all alphabetic characters in uppercase.

```
shell$ cat test0
abcde
zoolander - is a movie.
HELLO, WORLD!
123 ABC! def?
XYZ
UVWXYZ
shell$
```

```
shell$ cat output0
FGHIJ
ETTQFSIJW - NX F RTANJ.
MJQQT, BTWQI!
123 FGH! IJK?
```

```
BCD

XYZABC

shell$


shell$ cat output1

ABCDE

ZOOLANDER - IS A MOVIE.

HELLO, WORLD!

123 ABC! DEF?

XYZ

UVWXYZ
```