CSE 5A
Introduction to Programming I (C)
Homework 4

Read Chapter 7 **Due: Friday, October 26 by 6:00pm**

All programming assignments must be done **INDIVIDUALLY** by all members of the class. Start early to ensure you have enough time to submit your program before the deadline.

All assignments will be automatically collected from your computer account at precisely 6:00 pm on the due date shown above. These due dates and the time will be rigidly applied. An assignment that is not in place at 6:00pm on the due date will not be collected and therefore will not be graded, receiving a score of 0. If you have any questions/problems with the turn-in, see one of the tutors, TA, or the instructor.

All your programs must be neat and legible. Indent and comment your programs as shown in the text and as described in class. Make sure to thoroughly understand the assignment before you start on it. It is a good idea to verify your understanding of the problem with the tutors, TA, or instructor. For maximum credit, read and follow the style rules given in the **CSE 5A Program Grading** document (http://www.gregmiranda.com/wp-content/uploads/2018/08/CSE-5A-Program-Grading.pdf).

Getting Started

In the lab, log into the Windows computer using your cs5af account (not your UCSD account).

Using Visual Studio 2017, create a new project (File->New->Project…) and select the Visual C++ Empty Project. Change the **Name** of the project to HW4 (note the uppercase HW4). Change the **Location** of the project to your cs5af Home Directory (H:\). After changing the name and the location, click on the OK button.

In the Solutions Explorer window, create a new file by right clicking on the HW4 solution and selecting Add->New Item… from the context menu. Next, select C++ File (.cpp). Change the **Name** of the file to hw4.c. You do not need to change the location, it should default to your HW4 folder inside the HW4 project (H:\HW4\HW4\). After changing the name, click the Add button. Make sure to save everything (File->Save All).

Link to a video for creating a new project and source file: http://www.gregmiranda.com/cse5a-new-project

Feel free to ask the tutors on duty in the lab for help.

Program Description

Homework 4 (hw4.c) will be a program that implements a Caesar cipher. The program will also have functions, user input (**getchar()/scanf()**), conditional (**switch/if-else**), and looping (**while/do while** & **for** loop).

This program should be modular, with logical tasks broken up into functions. In this assignment you will write a program using functions to display output to the console (stdout). Do **NOT** use **global variables** in CSE 5A. Function headers are **required** for the function bodies. The function declarations should be placed before the **main()** function, while the function bodies should be placed after the **main()** function.

a) Your program must loop so that the user can continue to generate and view the output until the user enters the integer 0 to choose the **Exit** option from the menu (described later). NOTE: Your program will end **ONLY** with the input of **0**.
Use the following code as your **main()** driver:

```
int main(void)
{
    int   choice;               //Stores user input
    int   shift = 3;            //Number of letters to shift
    _Bool lowercase = 1;        //The case of the character to display

    menu();
    do
    {
        printf("\nEnter a number (0-5) to choose an option from the menu above: ");
        scanf("%d", &choice);
        getchar();

        switch(choice)
        {
            case 1:
                printCipher(shift, lowercase);
                break;
            case 2:
                encryptText(shift, lowercase);
                break;
            case 3:
                shift = readShift();
                break;
            case 4:
                lowercase = readCase();
                break;
            case 5:
                //encryptLine(shift, lowercase);
                break;
            case 0:
                printf("Exiting...\n");
                break;
            default:
                break;
        }
    } while (choice != 0);

    return 0;
}
```

b) Create a function called **menu()** with a return type of **void** which takes no parameters. Your function declaration should look like:

`void menu(void);`

In this function, you will display the menu of options to the user using **printf().**
The menu should appear to the user as:

```
Menu:
1) Print Cipher
2) Encrypt Text
3) Change Shift
4) Change Case
5) Extra Credit: Encrypt Line
0) Exit
```

c) Create a function called **printCipher()** with a return type of **void** which takes an **int** and a **_Bool** as parameters. Your function declaration should look like:

```
void printCipher(int, _Bool);
```

Use a loop to display the alphabet in the proper case (the **_Bool** parameter). The **_Bool** parameter is set to 1 (true) to display lowercase letters and 0 (false) to display uppercase letters. On the next line, use a loop to display the shifted alphabet in the proper case. To perform the shift, add the shift value to the character (the **int** parameter). Example with a shift value of 3: 'A' + 3 is 'D'. The shift value can also be negative, shifting the other direction. Example with a shift value of -3: 'A' - 3 is 'X'. Be careful with characters at the beginning or end of the alphabet that wrap around. Note: you will need to support numbers larger than 25 (and smaller than -25).

Calling **printCipher(3, 1)** produces the output (shift 3, lower case):

```
abcdefghijklmnopqrstuvwxyz
defghijklmnopqrstuvwxyzabc
```

Calling **printCipher(5, 0)** produces the output (shift 5, upper case):

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FGHIJKLMNOPQRSTUVWXYZABCDE
```

Calling **printCipher(29, 1)** produces the output (shift 29, lower case):

```
abcdefghijklmnopqrstuvwxyz
defghijklmnopqrstuvwxyzabc
```

d) Create a function called **readShift()** with a return type of **int** which takes no parameters. Your function declaration should look like:

```
int readShift(void);
```

In this function, take user input until a valid positive **OR** negative integer is entered (0 is neither positive nor negative).

Within the body of **readShift()**, create a variable called **shift** of type **int**. For example:

```
int shift;          /* Store user input */
```

Begin by prompting the user to enter a positive or negative integer by displaying the message **"Enter a positive or negative number to shift: "** to the console using **printf()**. Take user input using the **scanf()** Standard C Library function and store the result in **shift**. Display the error message **"ERROR: Enter a valid positive or negative**

number!" if zero is input. A loop will repeat the prompt until a valid input is entered. Return the **shift** value. Note: a value of 26 is valid, even though the result of shifting by 26 does not cause a shift of letters.

e) Create a function called **readCase()** with a return type of _Bool which takes no parameters. Your function declaration should look like:

```
_Bool readCase(void);
```

In this function, you will take user input until the user enters 'U' / 'u' or 'L' / 'l' character.

Within the body of **readCase()** create two variables called **character** and **lowercase**. For example:

```
char  character;          /* Stores user input */
_Bool lowercase;          /* Stores result of the input */
```

Begin by prompting the user to enter a 'U', 'u', 'L', 'l' by display the message **"Enter U for uppercase cipher or L for lowercase cipher [U/L]: "** to the consoles using **printf()**. Take user input using the **getchar()** Standard C Library function. Store the character entered by the user in the **character** variable. Check to make sure the user entered valid input. Display the error message **"ERROR: Enter U for uppercase cipher or L for lowercase cipher! "** if the character entered is not valid. Prompt for another character to be input. A loop will repeat the prompt until a valid input is entered. Once the user has entered valid input, save the result in **lowercase**. Return the **lowercase** value.

f) Create a function called **encryptText()** with a return type of **void** which takes an **int** and a **_Bool** as parameters. Your function declaration should look like:

```
void encryptText(int, _Bool);
```

Like **printCipher()** the **int** parameter holds the shift value and the **_Bool** parameter holds the lowercase flag. Ask the user to enter a word to encrypt by displaying the message **"Enter a word to encrypt: "** to the console using **printf()**. In a loop, take user input using the **getchar()** or the **scanf()** function. After reading each character, display the shifted value of the character to the console in the selected case from the **_Bool** parameter. You may need to convert an uppercase letter to lowercase or vice versa. Loop until the user enters a newline character '\n'. Note: any character that is not a letter should not be shifted and displayed as is (spaces, numbers, punctuation, etc.). Keep in mind that the loop doesn't run until the user hits the enter key, so nothing needs to be stored except for a single character that is read in, shifted, and then printed.

Calling **encryptText(3, 1)** could produce the following output:

```
Enter a word to encrypt: this is a test!
wklv lv d whvw!
```

Calling **encryptText(5, 0)** could produce the following output:

```
Enter a word to encrypt: this is a test!
YMNX NX F YJXY!
```

HINT: Solve the problem in small steps. Here's a suggestion.

1) Enter the code for step a), b), c). Test
2) Write the code for step d). Ignore error checking for now. Test.
3) Write the code for step e). Ignore range checking for now. Test.
4) Write the code for step f). Test.
5) Complete the code for step d) and e) above to range check your input. Test
6) Test. Test. Done!
7) Work on the extra credit!!

Extra Credit

Prior to working on the extra credit, create a backup of your file just in case something goes wrong. I recommend storing it on Google Drive or Dropbox.

Extra Credit Part 1 (+1 points)

g) Create a function called **convertChar()** with a return type of **char** which takes an **int**, **_Bool,** and **char** as parameters. Your function declaration should look like:

```
char convertChar(int, _Bool, char);
```

This function takes the **char** parameter and shifts it by the **int** parameter and changes the case to match the **_Bool** parameter, just as in **printCipher()**. If the character in the **char** parameter is not a letter, return the value as is (unshifted, no case change). Otherwise, return converted character (shifted, case changed).

Modify **encryptText()**to use **convertChar()** to shift the value and change the case prior to printing the value to the console. Make sure to remove the outdated code (or comment it out). After modifying **encryptText()**, the program should work the same way, just using the **convertChar()** function instead of having the shifting code inside the **encryptText()** function.

Extra Credit Part 2 (+1 points)

h) For this part, you will need to create two functions. The first function, **encryptLine()**, is used to call the second function, **encrypt()**, from the menu. Together, both functions work exactly like **encryptText()** from part f.

First, create a function called **encryptLine()** with a return type of **void** which takes an **int** and a **_Bool** as parameters. Your function declaration should look like:

```
void encryptLine(int, _Bool);
```

Like **encryptText()** the **int** parameter holds the shift value and the **_Bool** parameter holds the lowercase flag. Ask the user to enter a word to encrypt by displaying the message **"Enter a word to encrypt: "** to the console using **printf()**. Next, call the **encrypt()** function.

Create a function called **encrypt()** with a return type of **void** which takes an **int** and a **_Bool** as parameters. Your function declaration should look like:

```
void encrypt(int, _Bool);
```

This function is a recursive function, which is a function that calls itself. Every recursive function needs an end case, or it will act like an infinite loop until all the program's available stack space is used up and the program crashes. The end case for this function is the newline character '\n'. Each time the function is called, it should read a character from stdout using **getchar()** or **scanf()**. Then it should call the **convertChar()** function and print out the shifted value to the console. Finally, if the newline character hasn't been read in, the function should call itself.

Example Execution of the Program (user input is in bold):

```
Menu:
1) Print Cipher
2) Encrypt Text
3) Change Shift
4) Change Case
5) Extra Credit: Encrypt Line
0) Exit

Enter a number (0-5) to choose an option from the menu above: 1
abcdefghijklmnopqrstuvwxyz
defghijklmnopqrstuvwxyzabc

Enter a number (0-5) to choose an option from the menu above: 3

Enter a positive or negative number to shift: 5

Enter a number (0-5) to choose an option from the menu above: 1
abcdefghijklmnopqrstuvwxyz
fghijklmnopqrstuvwxyzabcde

Enter a number (0-5) to choose an option from the menu above: 4

Enter U for uppercase cipher or L for lowercase cipher [U/L]: k

ERROR: Enter U for uppercase cipher or L for lowercase cipher!

Enter U for uppercase cipher or L for lowercase cipher [U/L]: u

Enter a number (0-5) to choose an option from the menu above: 3

Enter a positive or negative number to shift: 0
ERROR: Enter a valid positive or negative number!

Enter a positive or negative number to shift: -5

Enter a number (0-5) to choose an option from the menu above: 1
ABCDEFGHIJKLMNOPQRSTUVWXYZ
VWXYZABCDEFGHIJKLMNOPQRSTU

Enter a number (0-5) to choose an option from the menu above: 2

Enter a word to encrypt: This is a test!
OCDN DN V OZNO!
```

```
Enter a number (0-5) to choose an option from the menu above: 5

Enter a word to encrypt: This is a test!
OCDN DN V OZNO!

Enter a number (0-5) to choose an option from the menu above: 9

Enter a number (0-5) to choose an option from the menu above: 0
Exiting...

Press ENTER to exit!
```

Final Notes

Make sure you have your name and cs5af account number in the header comment of hw4.c.
Note: replace "XX" in the Login: "cs5afXX" with your unique login name.

```
/*
 * Name: Jane-Joe LastName
 * Login: cs5afXX
 * Date: Month Day, Year
 * File: hw4.c
 * Sources of Help:

 * General description of the program ...
 */

#pragma warning(disable:4996)                          //Disable security warnings
#include <stdio.h>
```

Make sure to include the #pragma warning line after the program header or your program will not compile in Visual Studio.

Comment your program like the previous programs (and the Program Grading guide). Make sure to use meaningful comments.

You can download an example executable for Homework 4 from the following link:
http://www.gregmiranda.com/cse5a-hw4-sample

Verify you saved your work in the cs5af HOME directory (H:\HW4\hw4.c or H:\HW4\HW4\hw4.c).

For homework 4, we will also be turning them in using gradescope.com, which is an LMS (learning management system). Grade Scope is an online tool that will make it easier for us to collect homework assignments, grade them, and return them to you.

Keep an eye out for an email from Grade Scope as well as a Piazza post with instructions on how to turn in your homework 4. However, make sure to keep a copy of your homework 4 in the HOME directory like previous homework assignments just in case there is an issue.