CSE102 – Computer Programming (Spring 2021)
Homework #2

Handed out: 3:45pm March 11, 2021.

Due: 11:55pm March 22, 2021.

Hand-in Policy: Via Moodle. No late submissions will be accepted.
Collaboration Policy: No collaboration is permitted.
Grading: This homework will be graded on the scale of 100.

Description: In this homework, you will write a complete C program implement several functions as described below. You are expected to reflect what you have learned in class up to this point.

You are provided with six separate files (in HW2_Src.rar):

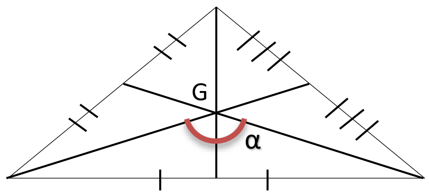
- **main.c:** Contains the main function. You are not expected to modify this file in your submission. You may modify it for your own testing and debugging needs.
- **hw2_io.h:** Contains the declarations of input and output related functions for this homework. You are not expected to modify this file in your submission. You may modify it for your own testing and debugging needs.
- **hw2_io.c:** This file will contain your implementation of the functions declared in the associated header file. The details of these functions' behaviors are provided below.
- **hw2_lib.h:** Contains the declarations of some more functions for this homework. You are not expected to modify this file in your submission. You may modify it for your own testing and debugging needs.
- **hw2_lib.c:** This file will contain your implementation of the functions declared in the associated header file. The details of these functions' behaviors are provided below.
- **makefile:** This is a makefile provided for you to use for compiling and testing your code.

The following provides the details of the functions to be implemented:

- **int find_WeekDayOfDate (int* day, int* month, int* year)**: Reads three numbers from the user input as a date and finds the corresponding day (Monday, Tuesday, etc.). Represent each day with a number 0 for Monday, 1 for Tuesday and so on.

- **int count_DayBetweenDates (int* starting_date, int* ending_date, int* date_of_year, int* day_of_week)**: Reads four parameters from the user as follows:
  starting_date (Use *ddmmyyyy* format e.g., 17031948),
  ending_date (Use same format in above e.g., 23081982),
  date_of_year (Use *ddmm* format only. Not input a year information e.g., 1809),
  day_of_week (0 for Monday, 1 for Tuesday etc.)
  The function should find how many times a specific date (e.g., 0902 as 9 February) corresponds a day (e.g., 2 as Wednesday) between starting (e.g., 23071930 as 23 July 1930) and ending dates (e.g., 10051991 as 10 May 1991).

- **double find_angle (double a0, double a1, double a)**: Reads three decimal numbers from the user representing length of the edges indicated in the figure below. Then, it calculates the angle shown as α in the figure. Display the calculated angle in digits as well as radians with two digits after decimal point.



- **void word_prediction (char* start_letter, char* end_letter, int* length_of_word, int* number_of_guess)**: Reads four parameters from the user as follows:
  start_letter and end_letter indicates range of letters in English alphabet that function can use for forming a word,
  (e. g. start_letter=c, end_letter=p the function can create the word: "code" and not create "chee**ta**h" since p < t and a < c)
  (e. g. start_letter=i, end_letter=t the function can create the word: "imposter" and not create "**fra**nkl**y**" since a & f < i and t < y)
  length_of_word indicates the length of word the function must form for,
  number_of_guess indicates how many times the user can make a guess to find out the word.

  The function randomly forms an English word within specified range of letters (English Alphabet) and the user tries to find out what the word is. After the function is called once, it should ask for user to input a letter and then it displays the found ones in the word. This process should continue until either number of rights of guess runs out or the word is found. Execution of the function can be designed as.

```
Please Input a letter: u
The word is: *u*u**

Please Input a letter: b
The word is: *ubu*b

Please Input a letter: s
The word is: subu*b

Please Input a letter: r
The word is: suburb
Congrats! You find the word.
```

  Note: As anyone can foresee, the word that the function generates most probably will not make sense. So, this task a kind of finding a character sequence.

- **char* string_matching (char* filename, char* searched_word, int* number_of_sentences)**: Reads three parameters from the user and searches exactly specified word in these sentences. You should prepare a file (with txt extension) containing sentences, paragraphs etc. Your function should take name of the file (filename) as parameter. If it finds any matching in one or more sentences, it should return that ones. number_of_sentences parameter refers to the first # (e. g. 5) sentences containing the specified word.

- **void encrypt (char* word)**: Write a function that encodes every character of a string as hexadecimal number according to any formula you choose and write it to text file named 'encoded.txt' as tab delimited. In 'encoded.txt', each encoded word should be located exactly in a single line.

- **char* decrypt (char* encoded)**: Write a function that takes encoded expressions from 'encoded.txt' and turns it to a string.

- **void curve_fit (char* filename):** Reads points (x, y) from 'points.txt' file and fits a curve for a second order polynomial equation using least squares method. At the end, it should display coefficients of the equation following. Display the coefficients as 3 digits after point.

$$f(x) = ax^2 + bx + c$$

Useful Hints: Here are some things that might make your development a bit easier.

- For testing your code use files for inputting data and getting the output. For example:
  ```
  $ hw2 < input.txt > output.txt
  ```
  will get the input from the file "input.txt" and will write the output to the file "output.txt". This way you can easily make a lot of entries to test your code without using the keyboard again and again.
- Use the makefile to compile your code. You can add a run case to your makefile to do the compilation and testing with one simple make command.

What to hand in: You are expected to hand in a zip or rar file including the six files above. Your implementations should be completed in "hw2_io.c" and "hw2_lib.c". The rest of the files should not be modified from their original versions.

- **HW2_lastname_firstname_studentno.rar / HW2_lastname_firstname_studentno.zip**