CSE102 – Computer Programming (Spring 2021)
Homework #5

Handed out: 11:55pm April 2, 2021.

Due: 11:55pm April 14, 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 a scale of 100.

Description: In this homework, you will write a complete C program that implements several functions as described below. You are expected to reflect on what you have learned in class up to this point. Therefore you are not allowed to use arrays. And also you should not utilize any library function except mentioned in this document. You can use any helper function as long as you construct it on your own.

You are provided with four separate files (in HW5_Src.rar):

- **hw5_main.c:** Contains the main function. You are not expected to modify this file in your submission except operate_polynomials function. You may modify it for your testing and debugging needs.
- **hw5_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 testing and debugging needs.
- **hw5_lib.c:** This file will contain your implementation of the functions declared in the associated header file. The details of the behavior of these functions 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:

- **void operate_polynomials (double* a3, double* a2, double* a1, double* a0, double* b3, double* b2, double* b1, double* b0, char operator)**: Write a function that reads two polynomials (at most degree 3) and applies a user-selected operator on these (via test functions in hw5_main.c). Allowed operators are addition, subtraction, and multiplication. The polynomials are entered with four coefficients. The function will return the resulting polynomial coefficients. User should input as (degree, coefficient). For equation $y = ax^3 + bx^2 + cx + d$, the input should be taken from the terminal as: (3, a), (2, b), (1, c), (0, d).
- **void four_d_vectors (double* mean_a0, double* mean_a1, double* mean_a2, double* mean_a3, double* longest_distance, int N)**: Write a function that takes N 4D vectors of 4 double numbers from the user by calling the following function inside this function;
    - **void distance_between_4d_points (double d0, double d1, double d2, double d3, double* euclidian_distance)**: This function should take the difference of two consecutive 4D vectors and find the euclidean distance between them.

  The last vector should be indicated with –1 –1 –1 –1. Find the average of each dimension across N vectors and return it as an output of the function. At the same time, calculate the longest Euclidean distance between two consecutive 4D vectors and return this value as the fifth argument.

- **void dhondt_method (int* partyA, int* partyB, int* partyC, int* partyD, int* partyE, int numberOfSeats)**: The D' Hondt method is a highest averages method for allocating seats in an electoral system and thus a type of party-list proportional representation. The method works like this; in each iteration, the initial vote of the party which has the highest number of the vote in that iteration is divided by the previous divider (initially 1) + 1, and that divided votes are stored for the next iteration. If you look closely at votes in the 2nd iteration, initial votes (100000) of the party which has the highest number of votes in that iteration (Party 1) are divided by the previous divider (which is 2) + 1 which is 3, and, that votes (33333) are stored for 3rd iteration. Notice that each party has own divider number (initially 1) that varies independently from each other.

Iteration	Party 1	Party 2	Party 3	Winner
Initial Votes	**100000**	80000	30000	Party 1
1	100000/2=50000	**80000**	30000	Party 2
2	**50000**	80000/2=40000	30000	Party 1
3	100000/3=33333	**40000**	30000	Party 2
4	**33333**	80000/3=26666	30000	Party 1
5	100000/4=25000	26666	**30000**	Party 3
6	25000	**26666**	30000/2=15000	Party 2
7	**25000**	80000/4=20000	15000	Party 1

  In this case, Party 1, Party 2, and Party 3 win 4, 3, 1 seats respectively. Write a function taking the count of votes of parties as input argument and return the number of seats they win by storing them in the parameter related to them. Print results in test functions of hw5_main.c.

- **void order_2d_points_cc (double* x1, double* y1, double* x2, double* y2, double* x3, double* y3):** For given three 2D points A(x1,y1), B(x2,y2) and C(x3,y3), order these points in counter-clockwise and return them in the correct order. You can use only cos () or sin () from math.h library.
- **void number_encrypt (unsigned char* number)**: By assuming that input is going to be between 0 and 255, encrypt the number by calling the following functions respectively inside this function;
  - **void get_number_components (unsigned char number, char* b7, char* b6, char* b5, char* b4, char* b3, char* b2, char* b1, char* b0):** Converts the number into binary and returns in bits. No bits operations are allowed. You can do only arithmetic operations.
  - **void reconstruct_components (unsigned char* number, char b7, char b6, char b5, char b4, char b3, char b2, char b1, char b0):** Constructs a new number according to the following algorithm and returns an unsigned char:
    Change bit-7 with bit-2, bit-6 with bit-3, bit-5 with bit-0, bit 4 with bit-1.
    7 6 5 4 3 2 1 0 --> 2 3 0 1 6 7 4 5

    Do exactly two circular left shift and obtain: 2 3 0 1 6 7 4 5 --> 0 1 6 7 4 5 2 3

Lastly, print out the number you obtain in main ().

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:
  ```
  $ hw5 < 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 your implementation in "hw5_lib.c".

- **HW5_lastname_firstname_studentno.rar / HW5_lastname_firstname_studentno.zip**