Reminder!

For all future communications, please use the following SUBJECT LINE:

CIS 27 Spring 2024 YourName : Needs/Questions

Without the above SUBJECT LINE, your emails will be deemed as SPAM/Phishing/Virus and will be deleted.

For all homework submission, please use the following SUBJECT LINE:

cis27Spring2024YourNameHw#.c

where **#** is replaced by the homework number; i.e., 1, 2, 3, etc.

Without the above SUBJECT LINE, your submission emails will be rejected, your homework is considered as not submitted.

Turn In:

Exercise #1 – Due on **Tuesday, February 13, 2024 by 11:00pm as Email Submission**

Homework Due Dates and Consideration:

All homework will be submitted by the given date and time (11:00pm). The homework submissions will be through emailing the work (programs and others) as attachments based on the specified and required formats and structures.

As submitting work through emails, there are time stamps for each email—please keep this in mind as you may want to check your work and submission.

If the submission is after 11:00pm of the given date, then it is considered as late. A late submission within 24 hours will get a 50 % penalty. Between 24 hours and 48 hours late, the penalty will be 75%. After 48 hours late, the submission will be ignored regardless of reasons and a zero (0) will be given.

There will be about 5 to 6 assignments for the semester. Some assignment will have 6-day time; some will have 10 days, and 2+ weeks as specified. Every student must start a homework as soon as it is posted on Canvas—students are responsible to check on Canvas for class information including the Canvas Homepage, Modules, Quizzes, etc.

A lot of hints for assignment are discussed in classes and code samples are posted on Canvas. Everyone will have the same length of time to submit regardless of reasons as a lot of hints are going to be provided. Start to work on homework/assignments early and all work must follow the instructions given in class.

Homework formats/conventions/styles are provided. These coding format, convention, and style must be followed to obtain full credit for all assignments. More information will be given through code demonstration in class — I do coding live during class meetings and discussions, and I use only Visual Studio IDE!

a) For each exercise, a package must be generated to include the following items:

- Copy of your source file (C program)—your source file must be named as

  `cis27Spring2024YourNameHw1.c`

- Copy of output (copy and paste to the end of your program in the PROGRAM_OUTPUT comment block as required)
- Copy of Logic_Code_Output_Issues (as a separate comment block) after the PROGRAM_OUTPUT block.

b) Emailing each package as follows,

- One email message for each exercise.
- The SUBJECT line of the message should have the following line:

  cis27Spring2024YourNameHw1.c

- Attaching the source file that was created in Part a).

Note 0!

"YourName" means FirstnameLastname; no abbreviation!

Note 1!

- For all functions that you are to write for the homework, you must append the intitals of "YourFirstName YourLastName" at the end of the function names—all functions. For examples, assuming the name is "**Last First**",

```
void displayClassInfoLF(void);
void runMenuFinalExamLF(void);
```

- Except for the indices of i, j, k, etc., all other variables must have the initials of your first name and last name added to the end of the variable names. For examples,

```
int usrInputLF;
int digitCountLF;
int absValueLF;
int tmpLF;
```

- You must declare all local variables at the top of the function; one declaration per statement except for the indices that you may need/have.
- Please be consistent with the above format requirements.
- Please follow all of the code formats, idioms, and practices that have been presented in class as well as in code and coding convention C file (posted on Canvas).
- Penalties will be applied for bad coding and practices as discussed/quizzed/demonstrated throughout class.
- Do not manually modify/change/insert text for the output of your code! You will get zero (0) for the work! I will run your code and know if you manually alter the pasted output as submitted in your C file!

Note 2!

1. We write code to manipulate data (which are provided by the user) to produce the required outcome in the most efficient way!
2. Getting the program to work is not enough to earn full credit. Your program must run correctly and follow all proper convention and consistent styles as explained in class in order to receive credit accordingly.

3. Writing code is not just the code works. It also involves patience, care and code idioms + forms along with others. Please see the posted code that have been written in class as well as the coding convention C file — posted on Canvas.

Note 3!

1. You will get zero (0) points if your code does not compile! Please make sure that you compile your code frequently and properly throughout the working session. Please check and run your submission again exactly as you just submitted.
2. You are only allowed to use four (4) functions of `printf()`, `scanf()`, `malloc()`, and `free()` defined in either `stdio.h`, or `stdlib.h`, or both. All other functions must be written by you!
3. A `runMenuHwNumberFL()` function has a menu that MUST BE a combination of `do-while` and `switch`. Any other form of the menu will receive 0 for the whole homework.
4. You will be penalized heavily if there are violations on code conventions as explained in class, homework submissions, quizzes, and document posted on Canvas!
5. Your code must work with all reasonable data sets/patterns. At least, your code should be tested with the given data samples indicated/given from the exam/document, etc.
6. Pay attention to naming; that means the specified/required names, your own generated names, and filenames. Penalty points will apply if you do not follow the instructions.
7. Again — You are only allowed to use `stdio.h` and `stdlib.h` for `printf()`, `scanf()`, `malloc()`, and `free()`. All other functions must be written by you!
8. All function names must have the initials of your firstname and lastname appended at the end.
9. The following functions are absolutely bad in my compilation — “cannot compile” error! Do not ever use them in my classes.

```
void doNotUseArraySyntax01() {
    int arySize1;

    printf("\nEnter an int for array size: ");
    scanf("%d", &arySize1);

    int ary2[arysize1];

    // TODO

    return;
}

void doNotUseArraySyntax02() {
    int arySize2 = 5;
    int ary2[arySize2];

    // TODO

    return;
}
```

10. Q.E.D.

++++++++++

For Fun – The following reminders were part of the final exam for my CIS 6 class. Please read and also check out the code convention!

We write code to manipulate data (which are provided by the user) to produce the required outcome in the most efficient way!

For Code Convention and Style –

Again, writing code is not just the code works. It also involves care, patience, coding idioms + forms, and other reminders. Please see the posted code written in class and the explanation of coding convention C/CPP file.

AND

If writing in C, you are only allowed to use printf() and scanf() from the stdio.h header. You will write everything else by yourself.

If writing in C++, you are only allowed to use cout and cin from the iostream header. You will write everything else by yourself.

ALSO,

- Including PROGRAM COMMENT block!
- Inserting required Comment lines!
- Removing unnecessary blank lines!
- If needed, 1 blank line is sufficient!
- Keeping lines to be reasonable short; for examples, around 65 characters!
- For functions without arguments, insert void in the function prototypes!
- For functions without arguments, remove void in the argument list of the function definitions!
- Keeping indentation levels with the same consistently!
- Declaring all variables at the top of function!
- Declaring one variable per line except for the indices!
- Using better name!
- Following the naming rules!
- Removing all comments that are not required!

- Inserting proper spaces around '(', ')', '{', and operators!

- Using proper code idioms, and efficient operators and styles!

- Paying attention to the exact layout of the required output to earn full credits!

- Replacing FL with the initial of your first-name and last-name!

For Operational Code -

- No global variables/values!

- No extern setups!

- Menu must be a combination of do - while and switch structures, and it has no arguments!

- Use initialization with proper initial value!

- If C++, you must use proper uniform initialization.

- Variables may not need to be initialized!

- If it is C++ coding, then
  "string" class is not allowed in this class. For the classes that are not written by you, only ostream (for cout) and istream(for cin) are allowed.
  You are only allowed to use cout and cin from the iostream header.

- Again, uniform initialization!

- All pointers must have values; that means initial address or setting address.

- After releasing the dynamic block, its pointer must be reset to NULL(in C) or nullptr (in C++).

- Dynamic objects and data->Memory management!

- In C, you are allowed to use only malloc() and free().

- In C++, you must use "new" and "delete" operators

- In C/C++, do not write

```
someVariable = -1 * somevariable;
someIncrement = someIncrement + 1;
if (someTestingValue != 0)
while (someTestingValue != 0)
```

- In C/C++, do write

```
        someVariable = -somevariable;
        someIncrement++;
        if (someTestingValue)
        while (someTestingValue)
```

- Use proper code idioms, and efficient operators and styles!

- Write your own code!

- Except for the member data, member functions, function arguments and local variables within member functions, and indices of i, j, k, etc., all other variables must have the initials of your first-name and last name added to the end of the variable names.

- For examples,

```
        int usrInputFL;
        int digitCountFL;
        int absValueFL;
        int tmpFL;

        // Function Prototypes
        void doGood(void);
        void displayDigit(int);

        // Function Definitions
        void doGood() {
            // Function Body
        }

        void displayDigit(int arg) {
           // Function Body
        }
```

- All stand - alone function names must have the initials of your first-name and last-name appended at the end.

- All filenames must have your complete first-name and last-name appended as required.

For OUTPUT and Copy of OUTPUT -

After copying the output to the driver, do not manually modify / change / insert text for the output of your code! You will get zero (0) for the work! I will run your code and know if you manually alter the pasted output as submitted in your C / C++ (and header) files!

For LOGIC_CODE_OUTPUT_Issues block -

Do not forget this block!

1. Coding Assignment

Exercise 1 – Due Tuesday, February 13 , 2024 by 11:00pm as Email

(1) Write a C program with call to functions to produce the output given below.

(2) The program should first display the output to screen (from function calls) as

```
We write code to manipulate data (which are provided by the user) to
produce the required outcome in the most efficient way!

CIS 27 - Data Structures and Algorithms
Laney College
Your Name

Information --
  Assignment:                HW #1 Exercise #1
  Implemented by:            Your Name
  Required Submission Date: 2024/02/13
  Actual Submission Date:    ____/__/__
```

Where "Your Name" means Firstname Lastname; no abbreviation!

For examples, if your name is **First Last** then Your Name should be First Last throughout all of your work/code as mentioned.

(3) The program will then continue to call other functions and display the results as follows,

```
// OUTPUT - Sample Run
We write code to manipulate data (which are provided by the user)
to produce the required outcome in the most efficient way!

CIS 27 - Data Structures and Algorithms
Laney College
Your Name

Information --
  Assignment:                HW #1 Exercise #1
  Implemented by:            Your Name
  Required Submission Date: 2024/02/10
  Actual Submission Date:    ____/__/__

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 6

Wrong Option!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 8
```

```
Calling displayDigitDetailFL() with one argument of
8

  While displayDigitDetailFL() is running with the actual
  argument of 8, the following is obtained:

  8 is an even and positive number.
  There is/are 1 digit(s).

  The digit(s) would be
    8

  There is/are 1 unique even digit(s) of
    8

  With digit 8 occurs the most time of 1!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 16494

Calling displayDigitDetailFL() with one argument of
16494

  While displayDigitDetailFL() is running with the actual
  argument of 16494, the following is obtained:

  16494 is an even and positive number.
  There is/are 5 digit(s).

  The digit(s) would be
    4
    9
    4
    6
    1

  There is/are 2 unique even digit(s) of
    4
    6

  With digit 4 occurs the most time of 2!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 42524502

Calling displayDigitDetailFL() with one argument of
```

```
42524502

  While displayDigitDetailFL() is running with the actual
  argument of 42524502, the following is obtained:

  42524502 is an even and positive number.
  There is/are 8 digit(s).

  The digit(s) would be
    2
    0
    5
    4
    2
    5
    2
    4

  There is/are 3 unique even digit(s) of
    0
    2
    4

  With digit 2 occurs the most time of 3!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 450466260

Calling displayDigitDetailFL() with one argument of
450466260

  While displayDigitDetailFL() is running with the actual
  argument of 450466260, the following is obtained:

  450466360 is an even and positive number.
  There is/are 9 digit(s).

  The digit(s) would be
    0
    6
    2
    6
    6
    4
    0
    5
    4

  There is/are 4 unique even digit(s) of
    0
    2
    4
```

```
    6

  With digit 6 occurs the most time of 3!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 44504006

Calling displayDigitDetailFL() with one argument of
44504006

  While displayDigitDetailFL() is running with the actual
  argument of 44504006, the following is obtained:

  44504006 is an even and positive number.
  There is/are 8 digit(s).

  The digit(s) would be
    6
    0
    0
    4
    0
    5
    4
    4

  There is/are 3 unique even digit(s) of
    0
    4
    6

  With digit 0 occurs the most time of 3!
  With digit 4 occurs the most time of 3!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 345

Calling displayDigitDetailFL() with one argument of
345

  While displayDigitDetailFL() is running with the actual
  argument of 345, the following is obtained:

  345 is an odd and positive number.
  There is/are 3 digit(s).

  The digit(s) would be
```

```
    5
    4
    3

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: -3450409

Calling displayDigitDetailFL() with one argument of
-3450409

  While displayDigitDetailFL() is running with the actual
  argument of -3450409, the following is obtained:

  -3450409 is an odd and negative number.
  There is/are 8 digit(s).

  The digit(s) would be
    9
    0
    4
    0
    5
    4
    3

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                          *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 41061

Calling displayDigitDetailFL() with one argument of
41061

  While displayDigitDetailFL() is running with the actual
  argument of 41061, the following is obtained:

  41061 is an odd and positive number.
  There is/are 4 digit(s).

  The digit(s) would be
    1
    6
    0
    1
    4

**************************************
*         MENU - Final Exam 0P       *
```

```
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                           *
**************************************
Enter an integer for option + ENTER: 1

Enter an integer: 0

Calling d displayDigitDetailFL() with one argument of
0

  While displayDigitDetailFL() is running with the actual
  argument of 0, the following is obtained:

  The given value is ZERO!

**************************************
*         MENU - Final Exam 0P       *
*  (1) Calling displayDigitDetailFL() *
*  (2) Quit                           *
**************************************
Enter an integer for option + ENTER: 2

Have a great Holiday Season!
```

At least, your program should have and use the following functions,

displayCodingStatementFL()

displayClassInfoFL()

runMenuFinalExamFL()

displayDigitDetailFL()

where YourName must be replaced by your first name and your last name. For examples,

If your name is **First Last** then YourName should be FirstLast throughout all of your work/code as mentioned. The initial pair will be "FL"

The sample run will have the options and values selected by the user. While testing, you must run your program to produce the required output.

(4) Save the program as **cis27Spring2024YourNameHw1.c**; and

(5) The above output should be copied and added to the end of the code in the PROGRAM_OUTPUT comment block. No manual manipulation to the output is allowed; you will get zero (0) for the whole work if any manual manipulation is found!

(6) Email the source code (your program) above using the SUBJECT LINE of

cis27Spring2024YourNameHw1.c