CIS1500 Assignment 3: Air Quality Index Generator

Total Marks: 10 marks
Weight: 10%
Due: Thursday, April 6th, 2023, at 11:59 pm

This assignment is built on A1 and A2 and involves the same functionality with some improvements! The largest changes in A3 are the addition of structures and reading and writing to files. We are providing some functionality from A2, so make sure to use the skeleton code provided on Courselink.

As stated in the course outline There will be **NO REGRADES** for this assignment.

1.2 Reminders

- Please remember to rename your files and the zipfile you submit correctly. **Incorrect naming will result in a grade of 0.** See specifics in Section 4.

- **3 FILES IN A SIGNLE ZIP: studentNumberA3_main.c A3_functions.c A3_functions.h**
  - studentNumberA3_main.c should be submitted with your student number;
  - A3_functions.c and A3_functions.h should not be renamed.
  - The zipfile should be named studentNumberA3.zip
    - Make sure you submit a SINGLE ZIP FILE with extension .zip
    - You are to do this through the command line using the following command. Make sure to replace studentNumber with your own student ID.

  zip studentNumberA3.zip studentNumberA3_main.c A3_functions.c A3_functions.h

- **Only zip three files into a single zip file – additional files/folders may result in a grade of 0** (the autograder ignores files generated upon zipping, like __MACOSX generated from a mac). See specifics in Section 4.
- You must test your assignment on the SoCS Linux server, and it will be similarly graded on the SoCS Linux server for consistency.
- **You will receive a 0 if your code includes global variables (variables declared outside of functions), goto statements, or if your code fails to compile without errors.**
- Your code MUST compile with the following command:

  gcc -Wall -std=c99 studentNumberA3_main.c A3_functions.c

- There will be **NO REGRADES** for this assignment.

Hint: The lecture sample code in studentStructs.zip gives an example of reading a file, calculating the average of each row, and saving file data to a structs stored in an array. While not identical, the material is very similar to this assignment.

1.3 Program Requirements

All the background information remains the same as previous assignments. See A1 and A2 outlines for more details. The overall flow of the main program is:

- A menu with the options to read data from a file (get location and pollutant data), display AQHIs for a date or location, and exit:

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
```

- On selecting read location data, you will need to prompt the user to enter three filenames and then parse the files for the pollutant concentrations (this is done in function readData below).

```
Welcome to the Air Quality Index Generator!

Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
3
Please enter the filename for the O3 csv file.
guelph_o3.csv
Please enter the filename for the NO2 csv file for Guelph.
guelph_no2.csv
Please enter the filename for the PM2_5 csv file for Guelph.
guelph_pm2_5.csv
```

- For displaying aqhis for a given location, you will be asked to select which type of AQHI and which location to print. Then a table with all dates for that location will be displayed.
- For displaying aqhis for a given date, you will be asked to select which type of AQHI and which date to print. Then a table with all locations for that date will be displayed.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
1

Choose your AQHI settings:
        1. Calculate Standard Conditions
        2. Calculate Seasonal Conditions
2
Please enter the name of a Location.
Toronto

AQHI for Seasonal conditions at Toronto:
-------------------------------------------------------------------------------------------------
| Date      | AQHI      | Health Risk | Health Message                                          |
-------------------------------------------------------------------------------------------------
| 2021-01-01| 1.00      | Low         | Ideal air quality for outdoor activities.               |
| 2021-01-02| 1.00      | Low         | Ideal air quality for outdoor activities.               |
```

- The entire code must loop and continue until the user exits the program. If the user reaches a maximum of 10 locations, you must not allow any new data to be added; however, the user can continue to print tables from this information.

- Notes:
    - Each struct requires 3 files to be read. To reach the maximum of 10 locations, 30 files would be required; if you choose to test this you will need to download additional files from the link found in Section 2.2.
    - Any files your code will be tested against will contain the same month of data (eg Jan 1 – Jan 31). We do not expect you to handle the error case where data for different months is provided.
    - Data will always be from the same month within a file i.e. the dates in the file will not be across two months
    - You will need to make sure the date exists in the files in case of a particular date being chosen in order to show any results.

2.1 Skeleton Code

You are provided with a zip file on Courselink called **"A3_Skeleton.zip"** which contains three files; you are **required use this skeleton code as the basis of your program**. You will find that we have provided you with many functions to get you started (Section 2.3). You are expected to use the functions as provided since there is some variation in the menu and printing the table – changing any of the provided functions could lead to unnecessary deductions when grading.

You are provided with the file A3_functions.h, which is the header file you will be using for A3 functions – it contains the 5 functions prototypes that you are required to write, and the functions prototypes for function we have provided for you. Do not change the names/arguments/return types of the functions. Do not change the name of this file. This file includes a new struct called Location that you are required to use for this assignment:

```
typedef struct Location{
        char locationName[MAX_STRING_SIZE];
        char dates[MAX_DATES][MAX_STRING_SIZE];
        int numDays;
        float pollutantAverages[3][MAX_DATES];

        float standardAQHIs[MAX_DATES];
        float seasonalAQHIs[MAX_DATES];
} Location;
```

You may add additional functions to the A3_functions.h file, but do not in any way modify the existing code. A3_functions.c contains the completed definitions for functions we are provided (Section 2.3), you are required to add the functions from Section 2.4.

2.2 Data Files

The pollutant data for each location is provided in three separate files, for example, guelph_no2.csv, guelph_ozone.csv and guelph_pm2_5.csv. You are provided with nine such files on Courselink to make it easier to test. The files are obtained from: http://www.airqualityontario.com/history/index.php and you are welcome to select more if you like.

You will store the data along with the three AQHI values (standard, warm and cool) for each day into a struct. The struct will be created in the readData function which uses readAPollutantFile to read the three files (O3, NO2 and PM2_5), readData then returns the initialized Location struct. **Note that 3 files are required to create a single struct. Reading all 9 provided files will create 3 separate structs.**

Each file contains metadata, including location name. After the metadata the file contains pollutant data. There are at most 31 rows of data each representing one day of pollutant readings. Each row contains 24 readings (one per hour) for the particular date i.e. there are 24 ozone readings per day for 31 days in the given file guelph_ozone.csv. You will be required to parse these files to store the location name, average pollutant values for each day, and track the number of days. Since there are 24 readings per day, once you have them read in, you will need to get the average – you are expected to do this using the required getAverage function described below! The file will be read and processed in the readAPollutantFile function described below.

In the below image you can see a portion of a data file. Lines 1 to 14 are metadata, and you need to extract the location name from line 4 (the rest of the metadata can be ignored). Starting on line 15 you can see the actual polution data for the location. Row 15 contains a station ID and pollutant type that can be ignored, then the date that you need to extract, followed by 24 measurements that you must obtain and store the average from.

```
 GNU nano 3.2                                    guelph_o3.csv

1
2  From,January 1, 2021 EST
3  To,January 31, 2021 EST
4  Guelph, (28028) Hourly, Ground-level Ozone (O3) Concentrations
5  Station,Guelph (28028)
6  Address,Exhibition St./clark St. W.
7  Latitude,43.551611
8  Longitude,-80.264167
9  Elevation, 330 metres
10 Air Intake Height,4 metres
11 Pollutant,Ozone (Ozone O3)
12 Unit, parts per billion (ppb)
13 Remarks, -999 for missing data. 9999 for invalid data.
14 Station ID,Pollutant,Date,H01,H02,H03,H04,H05,H06,H07,H08,H09,H10,H11,H12,H13,H14,H15,H16,H17,H18,H19,H20,H21,H22,H23,H24
15 28028,Ozone,2021-01-01,4,3,2,2,1,1,1,1,16,23,25,27,31,33,33,31,29,28,29,27,26,26,24,
16 28028,Ozone,2021-01-02,23,23,23,23,27,27,26,23,26,26,30,32,31,31,32,32,31,28,28,27,32,30,31,31,
17 28028,Ozone,2021-01-03,31,26,23,25,28,29,27,29,24,23,24,26,27,27,27,27,26,23,23,21,20,17,18,22,
18 28028,Ozone,2021-01-04,26,29,29,9999,30,31,31,31,32,31,30,29,29,29,27,28,27,26,23,18,14,13,13,14,
19 28028,Ozone,2021-01-05,12,7,5,11,13,11,11,8,6,6,9,13,14,17,18,20,25,25,23,21,22,20,21,24,
```

You are not required to submit these text files with your final submission, as we will be testing your code with new CSV files formatted the same way. You will, however, be required to take in the three file names from the user and ensure those files exist before trying to read them.

2.3 Provided Functions

You are given definitions for the following from A2 (with appropriate modifications) in the A3 Skeleton code:

- void printTableHead(char label[MAX_STRING_SIZE], char equationName[MAX_STRING_SIZE], char col1Name[MAX_STRING_SIZE], int numLocations);
    - Use this functions once when printing a table, it will print "AQHI for ____ conditions at ____" with the equationName and label substituted.
    - The equationName argument should be Standard or Seasonal
    - The label argument will be a location name or a date.
    - The function will then print:
        - A line of dashes.
        - The table column names with the first being the string col1Name
        - A line of dashes.
- void printTableRow(char label[MAX_STRING_SIZE], float aqhi);
    - This prints a row of data, including
    - label - a string for column 1 (either a location or a date)
    - aqhi – a float representing the AQHI value
    - The appropriate health risk level
    - The appropriate Health Message
- void printTableFoot();
    - The function will then print a line of dashes.
    - int printMenuGetSelection();
    - Prints the main menu, requests input from user (looping until valid input received)
    - Returns –1 if the user chooses to exit
    - Returns the input menu option of 1,2, or 3 otherwise.
- float roundAQHI(float aqhi);
    - As described in A2
- float standardAQHI(float O3, float NO2, float PM2_5);
    - As described in A2
- float coolAQHI(float NO2, float PM2_5);
    - As described in A2
- float warmAQHI(float O3, float NO2, float PM2_5);
    - As described in A2
- int readLine(char line[], int size, FILE* fp);
    - As described in class
- int splitLineIntoStrings(char line[], char words[][MAX_STRING_SIZE], char seperators[]);
    - As described in class

2.4 Required New Functions

You will be adding the following new functions to A3_functions.c. You must complete the function and use in your program (the prototypes are in A3_functions.h and are not to be changed):

- float getAverage(float pollutants[24]);
  - This function takes in an array of pollutants, storing 24 values for a particular pollutant.
  - This is used to calculate the average value for that pollutant, which is then returned as a float.
  - Note that if a negative value for concentration (ex; -9999) or the invalid value 9999 is present in the pollutants array, it must not be used in calculating the average.
- int readAPollutantFile(char filename[], char location[], float pollutantAverages[], char dates[][MAX_STRING_SIZE]);
  - Takes in a filename, a location to be updated, an array to store calculated average pollutant values per day and an array to store the dates to which the pollutant averages correspond
  - The filename should be a single word containing no spaces. (For example if the file contains the name as “Toronto Downtown” it be stored as “Toronto”).
  - Use the getAverage to calculate the average values
  - The function returns 0 if the file could not be opened, or the number of days of data read otherwise.
- struct Location readData();
  - Ask the user for three filenames and read each file using readAPollutantFile
  - Populate and return a struct storing the location name, average pollutant values for at most 31 days, the number of days the values are read in for and the corresponding dates and the AQHI value in standard, seasonal setting for each day into a struct.
  - You must use the date from the data file to determine whether to use warm or cool AQHI calculations.
  - Note that for three files must contain data for the same location, we will not be entering files with different dates. However, if given files for locations that do not match, your code must be able to recognize the incorrect file input and let the user know.
  - If the locations do not match, prompt the user to re-enter the filenames (see Sample Flow 3)
- void printTableByDate(Location locations[], int numLocations);
  - If numLocations is 0 print “No location data has been entered. Please enter new location and data and try again.” then return. Do not print out sub menu.
  - This function will ask whether the user wants Standard or Seasonal AQHI values. Make sure to verify correct input.
  - Ask what Date to search for.
  - Print the appropriate table head (see Section 2.3)
  - Search through all locations, and print the appropriate AQHI values for location on the selected date. Use the function in Section 2.3 to print the row.
  - Note: if a location has no matching dates do not print any AQHI values.
  - Note: If no locations have matching dates print “###-##-## not found.” in the table, substituting the selected date. (see Sample Flow 1 and 2)
  - The print the table footer (see Section 2.3)

- void printTableByLocation(Location locations[], int numLocations);
    - If numLocations is 0 print “No location data has been entered. Please enter new location and data and try again.” then return. Do not print out sub menu.
    - This function will ask whether the user wants Standard or Seasonal AQHI values and verify correct input.
    - Ask what location to search for.
    - Print the appropriate table head (see Section 2.3)
    - Search through all locations and print the appropriate AQHI values for all dates of the location. Use the function in Section 2.3 to print the row.
    - Note: If no locations have matching names print “LOCATION not found.” in the table, substituting the selected location. (see Sample Flow 4 and 5)
    - The print the table footer (see Section 2.3)

You must ensure the function signatures are identical to those provided here. You are allowed to add as many variables as needed for each function implementation within the function; however, do not add or remove arguments, do not change argument/return types, and do not modify the .h file. We will test these functions individually by calling them with known arguments/user input and test if the output is as expected.

2.5 Optional Functions

You may choose to add other functions to your program as well. There is a dedication spot in A3_functions.h for you to add prototypes, and you may put the definitions in A3_functions.c. This is completely optional, and you will not be graded on these (so long as the functions from Section 2.4 are included and work).

You are permitted to use functions/code from lecture material for these, but you must reference where you obtained them in the function comment. Ex:

```
// reference: splitLinesIntoStrings function  obtained from lecture W9L3 slide 23, and fileIO.zip
```

2.6 Grading Breakdown

Your program flow will be worth 20% of this assignment.

Refer to the sample flows provided for how your program is expected to run (Section 3 Sample Flows).

Your required functions will be worth 50% of this assignment.

We will unit test each of the required functions and compare the results of your functions to those expected. For each function we may run multiple tests.

Your output text formatting will be worth 10% of this assignment.

We will flag differences in spellings, spaces, tables, new lines, etc. and test against different input values using the auto-grader, so make your you follow the output exactly. For each unique error, 0.5 marks will be deducted. The formatting grade will not be negative, it will only range from 0 to 1.

Refer to the sample flows provided for how your program is expected to run (Section 3 Sample Flows). Altering/customizing sentences will not be accepted, and you will receive a zero in this category if you do so.

Style: Indentation, variable names, and comments are worth 20% of this assignemnt.

Comments are most commonly used to explain confusing or not easily understood parts of code or for depicting that the following code carries out a certain piece of logic. These are also used to explain the program and add a header to it. A file header comment is a type of comment that appears at the top of each file before the #include line(s). **Comment grades include header comments (see 4.3 File Header Comment Format) and meaningful comments throughout your code.**

Each code block should be **indented for readability** (if/else statements are an example of a code block). We are looking for **meaningful variable names** that depict the values they are representing – this will make it easier for you to remember what each variable is for and for us to grade you!

1.4 Sample Flows

Note that **an auto-grader will either fully or partially grade your submission**. To get full grades, you need to match our output requirements exactly.

3.1 Sample Flow I

This sample flow takes place after reading all provided files (guelph, toronto, and toronto_aug). The AQHI values for the date 2021-01-10 are printed, this means that no data from the files/struct related to Toronto in August are printed.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
2

Choose your AQHI settings:
        1. Calculate Standard Conditions
        2. Calculate Seasonal Conditions
1
Please enter a date.
2021-01-10

AQHI for Standard conditions at 2021-01-10:
-------------------------------------------------------------------------------------------------
| Location  | AQHI      | Health Risk | Health Message                                          |
-------------------------------------------------------------------------------------------------
| Guelph    | 2.00      | Low         | Ideal air quality for outdoor activities.               |
| Toronto   | 3.00      | Low         | Ideal air quality for outdoor activities.               |
-------------------------------------------------------------------------------------------------

Select an option from the Menu:
```

3.1 Sample Flow 2

This sample flow takes place after reading all provided files (guelph, toronto, and toronto_aug). The user selected to print values for the date “2024-01-10” which does not exist.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
2

Choose your AQHI settings:
        1. Calculate Standard Conditions
        2. Calculate Seasonal Conditions
2
Please enter a date.
2024-01-10

AQHI for Seasonal conditions at 2024-01-10:
-------------------------------------------------------------------------------------------------
| Location  | AQHI      | Health Risk | Health Message                                          |
-------------------------------------------------------------------------------------------------
2024-01-10 not found.
-------------------------------------------------------------------------------------------------

Select an option from the Menu:
        1. Display AQHIs for Location
```

3.1 Sample Flow 3

This sample flow the user enters a valid filename for O3 data in Toronto, then a filename for a different location's NO2 data which requests the user to re-enter the data for Toronto. After entering a valid filename for NO2 data, the user enters a filename that does not exist for PM2.5 which causes the readAPollutantFile function to return 0 and print an error, the user is prompted for the filename again.

Note that after the first file is read, the name of the location is printed in the prompt for a filename.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
3
Please enter the filename for the O3 csv file.
toronto_o3.csv
Please enter the filename for the NO2 csv file for Toronto.
guelph_no2.csv
Please enter the filename for the NO2 csv file for Toronto.
toronto_no2.csv
Please enter the filename for the PM2_5 csv file for Toronto.
sldikrjtng.csv
ERROR: File sldikrjtng.csv does not exist
Please enter the filename for the PM2_5 csv file for Toronto.
toronto_pm2_5.csv

Select an option from the Menu:
        1. Display AQHIs for Location
```

3.1 Sample Flow 4

This sample flow takes place after reading data for both Guelph and Toronto in January. The user selected to print location "nowhere" which does not exist.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
1

Choose your AQHI settings:
        1. Calculate Standard Conditions
        2. Calculate Seasonal Conditions
1
Please enter the name of a Location.
nowhere

AQHI for Standard conditions at nowhere:
-----------------------------------------------------------------------------------------------
| Date      | AQHI      | Health Risk | Health Message                                        |
-----------------------------------------------------------------------------------------------
nowhere not found.
-----------------------------------------------------------------------------------------------

Select an option from the Menu:
        1. Display AQHIs for Location
```

3.1 Sample Flow 5

These two images are from a single table after reading in the files for Toronto in both January and August. The data is printed in the order in which the files were read, then by date. So if the user had read the files for August the table would start 2021-08-01, 2021-08-02, etc. before printing the dates 2021-01-01, 2021-01-02, etc.

```
Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
        3. Read Data From a File
        4. Exit
1

Choose your AQHI settings:
        1. Calculate Standard Conditions
        2. Calculate Seasonal Conditions
2
Please enter the name of a Location.
Toronto

AQHI for Seasonal conditions at Toronto:
-----------------------------------------------------------------------------------------------
| Date      | AQHI      | Health Risk | Health Message                                        |
-----------------------------------------------------------------------------------------------
| 2021-01-01| 1.00      | Low         | Ideal air quality for outdoor activities.             |
| 2021-01-02| 1.00      | Low         | Ideal air quality for outdoor activities.             |
| 2021-01-03| 1.00      | Low         | Ideal air quality for outdoor activities.             |
| 2021-01-04| 2.00      | Low         | Ideal air quality for outdoor activities.             |
| 2021-01-05| 2.00      | Low         | Ideal air quality for outdoor activities.             |
```

…

```
| 2021-08-14| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-15| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-16| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-17| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-18| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-19| 6.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-20| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-21| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-22| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-23| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-24| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-25| 6.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-26| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-27| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-28| 4.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-29| 6.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-30| 5.00      | Moderate    | No need to modify your outdoor activities.            |
| 2021-08-31| 4.00      | Moderate    | No need to modify your outdoor activities.            |
-----------------------------------------------------------------------------------------------

Select an option from the Menu:
        1. Display AQHIs for Location
        2. Display AQHIs for Date
```

1.5 Program Submission and Administration Information

The following section outlines what is expected when submitting the assignment and various other administration information with respect to the assignment. To submit your assignment, upload a single zip file to the Dropbox box for A3 on Courselink.

4.1 The Submission File

The following is expected for the file that is to be submitted upon completing the assignment:

- You are to submit **one zip file containing studentNumberA3_main.c, A3_functions.h and A3_functions.c file** for your program.
  - Do not include any other files or folders in your submission; they will be ignored.
  - Do not include this file inside another folder; it will be ignored by the auto-grader.
- The name of the **zip file** follows the following format: **studentNumberA3.zip**
  - Example: John Snow's student number is 1770770 then the zip file name is 1770770A3.zip
- The name of the **C file** follows the following format: **studentNumberA3_main.c**
  - Example: John Snow's student number is 1770770 then the C file name is 1770770A3_main.c
- The name of the **C file** follows the following format: **A3_functions.c**
  - Do not change the file name from the skeleton.
- The name of the H **file** follows the following format: **A3_functions.h**
  - Do not change the file name from the skeleton.
- **Incorrect zip file, H file, or C file names will result in a grade of 0.**
- To ensure you zip correctly and in the right format, we require you to zip your submission through the command line using the following command:

  zip studentNumberA3.zip studentNumberA3_main.c A3_functions.c A3_functions.h

Make sure to replace studentNumber with your own student number.

You may wish to download your own submission from dropbox, transfer a copy back to the SoCS server and re-test after submission. This way you can be 100% certain you have submitted the correct documents.

4.2 Program Expectations

Your program is expected to follow the outlined information exactly. Failure to do so will result in deductions to your assignment grade.

1. Your program should be compiled and tested on the SoCS Linux server.
2. You must use the following command to compile your code:

   gcc -Wall -std=c99 studentNumberA3_main.c A3_functions.c -lm

   a. Example: For John Snow's submission, the command would be:
   gcc -Wall -std=c99 1770770A3_main.c A3_functions -lm
3. The program file you submit must compile with no errors
   a. **Programs that fail to compile will be given a mark of zero.**
4. Programs that produce warnings upon compilation will receive a deduction of 1 mark for each type/category of warning
5. The program files must contain instructions for the TA on how to compile and run your program in a file header comment (see 3.3 File Header Comment Format).

4.3 File Header Comment Format

Note: This sample uses the same John Snow example; the file name, student name, student ID, etc., and file descriptions must be changed per student.

```
/*************************** 1770770A3_main.c ***************************

Student Name: John Snow                          Student ID: 1770770

Due Date: Friday, 10th February 2023, at 11:59 pm    Course Name: CIS*1500

I have exclusive control over this submission via my password.

By including this statement in this header comment, I certify that:

1) I have read and understood the University policy on academic integrity; and

2) I have completed assigned video on academic integrity.

I assert that this work is my own. I have appropriately acknowledged any and all material (code, data, images, ideas or words) that I have used, whether directly quoted or paraphrase. Furthermore, I certify that this assignment was prepared by me specifically for this course.

*****************************************************************

This file contains…

Describe the program, functions, important variables, etc.

*****************************************************************

The program should be compiled using the following flags:

-std=c99

-Wall

-lm

Compiling:
```

gcc -Wall -std=c99 1770707A3_main.c A3_functions.c -o A3 -lm

OR

gcc -Wall -std=c99 1770707A3_main.c A3_functions.c -lm

Running the program:

./A3

OR

./a.out

**********/