CS2313 Homework 2: Fuel Efficiency Logger

Due Date: Thursday, September 26th, by 11:59 pm

Objective:

This assignment is designed to strengthen your skills with selection structures (`if/else`) and `while` loops. You will create a C++ program that logs trip data and computes various fuel-efficiency statistics by separating the data logic into a class and the user interaction into `main`.

Part A: The `MPGStats` Class

You will define a class named `MPGStats` in a header file, `MPGStats.h`. This class will be the "engine" of our program, responsible for storing and managing all the fuel efficiency data.

Private Data Members

Your class must contain the following private members to store its state:

- `int trips`: The total count of valid trips recorded.
- `double totalMiles`: A running total of all miles driven.
- `double totalGallons`: A running total of all gallons used.
- `int lowMPGCount`: The number of trips with an MPG below 15.0.
- `double bestMPG`: The highest single-trip MPG recorded.
- `double worstMPG`: The lowest single-trip MPG recorded.

Public Member Functions

Your class must provide the following public functions to interact with its data:

- **`MPGStats()` (Default Constructor)**
  - Initializes all numeric counters and accumulators to `0`.
  - **Important:** Initialize `bestMPG` to a very low value (e.g., `0.0`) and `worstMPG` to a very high value to ensure the first valid trip correctly sets them.
- **`double addTrip(double miles, double gallons)`**
  - This is the core function for updating the stats.
  - **Validate Input**: First, check if `gallons > 0` and `miles >= 0`.
  - If the input is **invalid**, do not update any member variables and return `-1.0`.
  - If the input is **valid**:
    1. Calculate the MPG for this trip (`miles / gallons`).
    2. Update all member variables: increment `trips`, add to `totalMiles` and `totalGallons`, increment `lowMPGCount` if `MPG < 15`, and update `bestMPG` and `worstMPG` if necessary.
    3. Return the calculated MPG for this trip.

- **Getter Functions**
  - Provide simple "getter" functions for each private member: `getTrips()`, `getTotalMiles()`, `getTotalGallons()`, `getLowMPGCount()`, `getBestMPG()`, and `getWorstMPG()`.
- **`double overallMPG() const`**
  - Calculates and returns the overall miles per gallon (`totalMiles / totalGallons`).
  - To prevent division by zero, it should return `0.0` if `totalGallons` is `0`.

Part B: The Main Program (`main.cpp`)

Your `main.cpp` file will handle all user interaction. It will create an `MPGStats` object and use its public functions to process data. The program flow has two distinct phases.

Phase 1: Counter-Controlled Loop

1. Ask the user how many trips (`T`) they intend to log.
2. Use a `while` loop that executes exactly `T` times.
3. Inside the loop, prompt the user to enter the miles and gallons for one trip.
4. Call the `addTrip` function. If it returns `-1.0`, print a brief error message and re-prompt the user for the same trip's data until it is valid.
5. On a successful `addTrip`, print the returned per-trip MPG, formatted to two decimal places.

Phase 2: Sentinel-Controlled Loop

1. Immediately after Phase 1, begin prompting the user for more trips.
2. Use a `while` loop that continues as long as the user does not enter `-1` for the miles. The value `-1` is the **sentinel** that signals the end of input.
3. For any other value for miles, read the corresponding gallons and process the trip using `addTrip` just as you did in Phase 1 (including validation and re-prompting).

Final Summary Output

After the user ends Phase 2 (by entering -1), print a final summary report containing:

- Total number of trips
- Total miles driven
- Total gallons used
- Overall MPG
- Best trip MPG
- Worst trip MPG
- Number of trips with low MPG (< 15)

All floating-point numbers in the output must be formatted to two decimal places.

Technical Requirements

- You must use `if/else` statements for selection and logical operators (`&&`, `||`, `!`) as needed.
- Use compound assignment operators like `+=` and the increment operator `++`.
- **Error Handling**: Your program must gracefully handle non-numeric input. If the user enters text where a number is expected, clear the stream's error state and re-prompt for input.

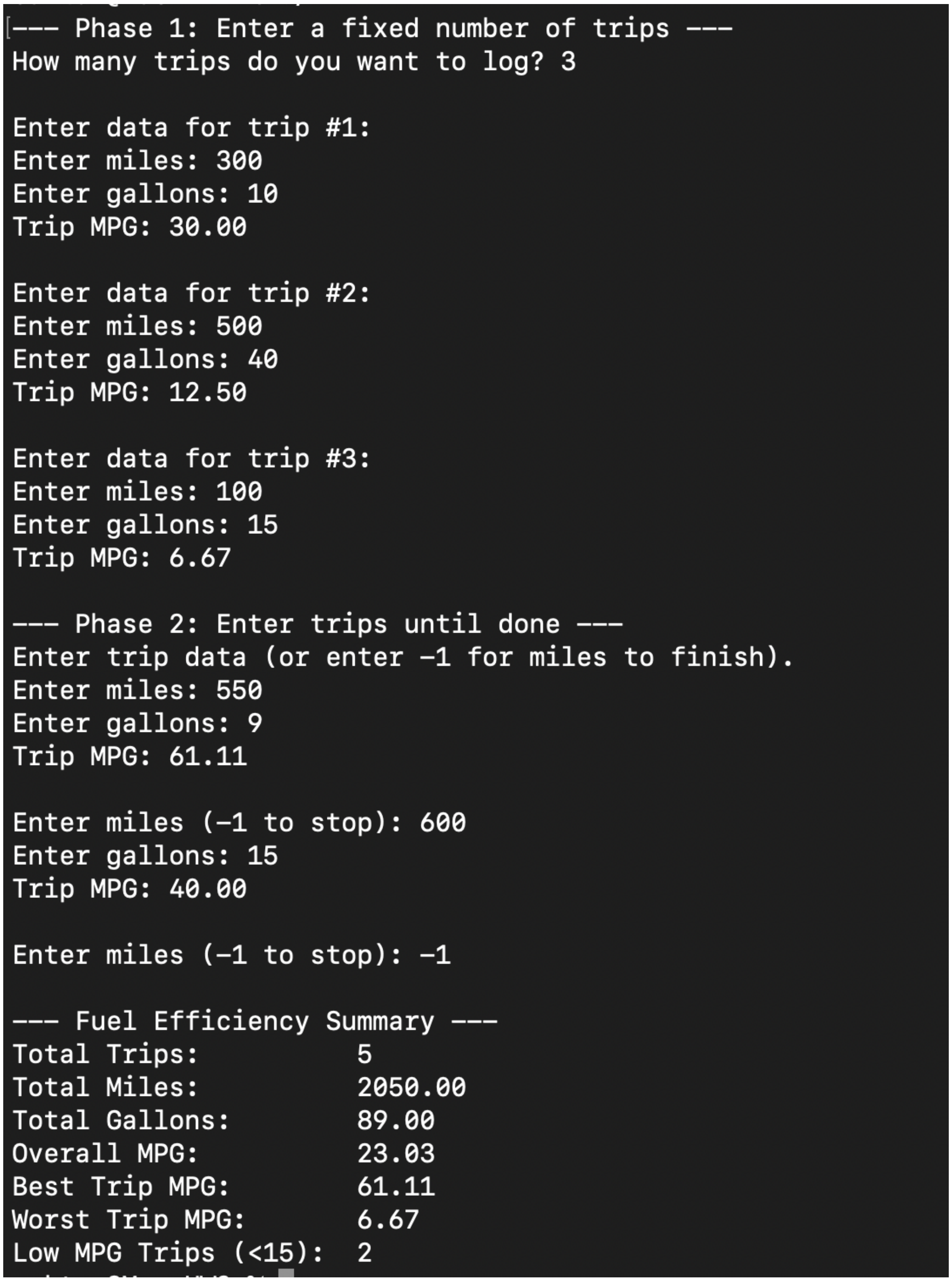
Submission Guidelines

1. Add the following header comment block to the top of both `MPGStats.h` and `main.cpp`:

```
// Name: Your Full Name
// Student ID: YourID
// Assignment: CS2313 Homework 2
```

2. Create a **`.zip`** archive containing the following files:
    - `MPGStats.h`
    - `main.cpp`
    - A screenshot of a single, complete program run. The screenshot must show:
        - Phase 1 running with at least 3 trips (`T >= 3`).
        - At least one instance of invalid trip data being rejected and then corrected by the user.
        - Phase 2 ending after the user enters `-1`.
        - The complete final summary report.
3. Name the archive file in the format: `Lastname_hw2.zip` (e.g., `Smith_hw2.zip`).

Sample Run:



```
[--- Phase 1: Enter a fixed number of trips ---
How many trips do you want to log? 3

Enter data for trip #1:
Enter miles: 300
Enter gallons: 10
Trip MPG: 30.00

Enter data for trip #2:
Enter miles: 500
Enter gallons: 40
Trip MPG: 12.50

Enter data for trip #3:
Enter miles: 100
Enter gallons: 15
Trip MPG: 6.67

--- Phase 2: Enter trips until done ---
Enter trip data (or enter -1 for miles to finish).
Enter miles: 550
Enter gallons: 9
Trip MPG: 61.11

Enter miles (-1 to stop): 600
Enter gallons: 15
Trip MPG: 40.00

Enter miles (-1 to stop): -1

--- Fuel Efficiency Summary ---
Total Trips:          5
Total Miles:          2050.00
Total Gallons:        89.00
Overall MPG:          23.03
Best Trip MPG:        61.11
Worst Trip MPG:       6.67
Low MPG Trips (<15):  2
```