CS262 Project 2: Bike Share Operations
Due 11:59 P.M., Tuesday Nov. 5, 2019

Introduction:

You are the president of a software company. The CEO of GMU's bike share company has contacted you with a proposal. He wants your company to build a new computer system to track and analyze bike trip operations. He will send you a data set that contains bike trips made in one day for you to use to create a prototype. The computer system you design will need to process this data.

The bike share company has two types of ***membership*** payment plans. A ***casual*** rider is not registered and pays for one bike trip at a time. A ***registered*** rider signs up for a longer term and pays a flat annual rate for unlimited rides at a lower cost. Each trip record contains the start and end bike share stations. A ***start station*** is a location where riders check a bike out of a docking station when they begin a bike trip. An ***end station*** is a location where riders will check a bike into a docking station when they end a trip. Each record also contains some time variables that you can use to compute the time and length of each trip. The records will contain the ***hour*** that the trip started in a 24-hour clock (military time), and the ***minutes*** that have passed during that hour. Example: hour = 16 and minutes = 30 means that trip started at 4:30 pm. There is also a ***duration*** variable that contains the length of time of each trip in milliseconds. And last but not least, each record contains the ***ID*** number of the bike that was ridden in the trip.

In order for the CEO to run his bike share operations, your system will need to process the data and provide him with the following types of information:

- The busiest time of day, when there are the most bike trips
- The busiest stations
- The slowest stations
- The average time of all the bike trips
- The longest and shortest bike trips

The CEO also needs to monitor the inventory of bikes and set up a quality control process to identify the bikes that get the most use so that they can be flagged for maintenance. He has asked you to create an inventory list of all the bikes and track some key metrics on each bike. He will need to know which bikes are getting the most use and where they are located at the end of the day.

Specifications:

You are given a text file with data on trips. You will need to read this data into an array. Each row will contain one trip struct with the definition provided below.

```
typedef struct Trip_struct{
  char membershipType[12];
  int startStationId, endStationId, bikeId,
     duration, startHr, startMin;
}Trip;
```

The order of the fields in each row is

MembershipType	StartStationId	EndStationId	BikeId	Duration	Hour	Minute

Here are the specifications for your system:

1. Information for each trip will be kept in a single data type (struct) and will consist of:
    - A bike ID (integer)
    - A membership Type (char string of 12 characters)
    - A start station ID (integer)
    - An end station ID (integer)
    - The duration of the trip in milliseconds (integer)
    - The hour in which the trip started in military time (integer)
    - The minute at which the trip started (integer)

2. The trip data will be read from an ASCII file at the start of the program. The data fields will be those listed above, separated by white space. Each line in the file will contain a single struct.
3. Use malloc to allocate space to store the trip array.
4. After reading in the data, your system will need to provide menus that the users of your system will navigate to obtain metrics.
5. The metrics that users will need are as follows:
    - Average number of trips per hour
    - The hour with largest number of trips
    - A report of the count of trips in each hour
    - Average duration of the trips (in both milliseconds and minutes)
    - The trip with the longest duration (in both milliseconds and minutes)
    - The 5 busiest starting stations (that had the largest number of trips in the day)
    - The 5 busiest ending stations (that had the largest number of trips in the day)
    - The percentage of trips in each membership type

To create the inventory of bikes, you will use a bike struct with the definition provided below.

```
typedef struct Bike_struct{
  char maintenanceFlag;
  int bikeId, endStationId;
  int numTrips, totalDuration;
  float totalMinutes;
}Bike;
```

You will need to create a second separate array of bike structs by implementing the following data structure and functions.

1. Information for each bike will be kept in a single data type (struct) and will consist of:
    - A bike ID, each struct will have a unique bike ID (integer)
    - A maintenance flag (character)
        - ‘0’ default value
        - ‘1’ when flagged for maintenance
    - An end station ID (integer)
    - The number of trips (integer)
    - The total duration of all the trips for this bike (integer)
    - The total minutes of all the trips for this bike (float)

2. You need to create an array of bike structs. Make the size of the array 4,000 elements. This will provide some empty rows for adding new bikes to the inventory.
3. Creating a list of unique bike IDs can be done in the following steps:
    - Sort the trip array by bike ID using a quick sort algorithm

- Create an empty bike array of 4,000 bike structs
- Write a function that loops through the trip array, copying only unique bike IDs into the bike array
- Write a function that loops through the trip array and updates the rest of the data members in each bike struct

4. The CEO of the bike share company has requested the following metrics on the bike inventory:
   - The 10 bikes with the longest duration in minutes
   - Where the 10 bikes above are located at the end of the day
   - The 10 bikes with the most trips
   - How many bikes were only used for 2 or less trips
   - For the bikes with 2 or less trips, make a list of the start station IDs and the number of trips for each of these stations

5. The CEO has also requested that your system is able to add and remove bikes from the inventory list

Implementation:

The specifications for this project have a great deal of detail. However, the fact that they are detailed can be used to your advantage, in that you can break the project into smaller units and work on them separately to create the complete project.

1. You will implement this program using arrays of structs. You will read data from an input text file into an array. Then you will use the data in this array to create a second separate array of 4,000 bike structs.
2. You will need to be able to sort the arrays by any given data member using quick sort.
3. Your program will be menu driven. As you create and work through your menus, use stubs as place holders until you write the code. The program will contain a main menu, and several choices from this menu will bring up a sub-menu. See the Menu Specifications below.
4. An input file containing the trip data is provided.
5. A sample script and sample output will hopefully be provided that you may test with your program (depending upon time). If provided, your program must respond to inputs in the same manner that the sample script responds. In other words, your program should not add unnecessary prompts for user input, nor should it skip prompts for input (or prompt for multiple data items with a single prompt). If the script cannot be provided in a timely manner (approximately one week before the due date), then this requirement will not be necessary.

Menu Specifications:

- **Main Menu**: The main menu will contain the following choices and functionality:
  - Read Trip Data
    - Prompt the user for a file name containing the trip data.
    - Open this file and read the information from the file, creating an array of Trip structs.
    - If the file cannot be found, print a message stating this fact, and return to the Main Menu.
  - Run Metrics
    - Display the Metrics Menu (See below)
  - Print Menu
    - Display the Print Menu (See below)
  - Flag Bikes for Maintenance
    - Make a list of the 10 bikes with the longest duration
    - Update the maintenance flag to ‘1’
  - Update Bike Inventory
    - Add a bike
    - Remove a bike
  - Quit
    - Exit the program

- **Metrics Menu:**
    - The user is presented with a list of metrics from the specifications above
        - The program prompts the user to choose a metric.
        - The program prompts the user to choose whether to display the metric on the screen or write the result to an output file.
        - If an output file is chosen, the program prompts the user to type in the name of the output file.
    - Return to Main Menu
- **Print Menu:**
    - Print All the Trip Data
        - The program prompts the user to choose whether to display the data on the screen or write the result to an output file.
        - If an output file is chosen, the program prompts the user to type in the name of the output file.
        - Return to Main Menu
    - Print All the Bike Inventory
        - The program prompts the user to choose whether to display the data on the screen or write the result to an output file.
        - If an output file is chosen, the program prompts the user to type in the name of the output file.
        - Return to Main Menu
    - Print the CEO Report
        - The program prompts the user to type in the name of the output file.
        - All the metrics for the trips and the bikes are printed to the output file in a nicely formatted style that is easy to read.
        - Return to Main Menu
    - Print one Bike
        - The program prompts the user to enter a bike ID
        - The program prompts the user to choose whether to display the data members for the bike on the screen or write the result to an output file.
        - If an output file is chosen, the program prompts the user to type in the name of the output file.
        - Return to Main Menu

Additional Specifications:

- Your program will have one source file
    - The source file will be named Project2_main_<username>_<lab_section>.c
    - This file will contain your main() function as well as the functions to handle the menus
- You will provide a Makefile that will compile your source file
- You may not use global variables.

Submission Instructions:

You will submit a tar file containing your source code files, your input text file, and Makefile. To do this, perform the following steps:

1. On zeus, make a directory to hold your files for submission. This directory must be named "Project2_<username>_<lab_section>"
2. Copy or move your source file, input text file, and Makefile to this directory.
3. Perform the following command to create your tar file:

    tar -cvf Project2_<username>_<lab_section>.tar Project2_<username>_<lab_section>

Submit this tar file to Blackboard for the Project 2 assignment.

Suggested Implementation Steps:

Below is a step by step guide you can use to create your program for this project. You do not have to follow this procedure, but you will likely have greater success in completing your project on time if you do so. Breaking the creation of the program into smaller portions will make your code easier to maintain and debug.

1. Create separate void functions with no parameters to print each menu (one for the Main Menu, one for the Metrics Menu, one for the Print Menu, etc.). Use these functions only to print the menus (do not prompt for a menu choice in these functions).
2. Create the logic to use the menus.
    A. In the main() function, create a loop to print the Main Menu (by calling its print function) and prompt for user input.
    B. Use switch statements based on user input to call other functions (don't put other functionality in the switch statements - just call other functions).Also break out of the loop if the user chooses to end the program.
    C. Create the other functions based on user input.
        1. If the function is to print a submenu, add functionality print the submenu, prompt the user for input, then add a switch statement to call "stub functions" based on user input.
        2. If the other functions are not submenu functions, create a "stub function." A stub function has no parameters or functionality other than to print an informational message, such as "In Function FindMaxDuration." You will add parameters and functionality to it later.
3. Test the logic of your menus by checking each combination of input from the Main Menu through the submenus and stub functions to ensure that each stub function is reached correctly.
4. Once you have the menus working, work on implementing the functions.

5. Test your code thoroughly

A suggested timeframe to complete the implementation steps is as follows:
Week 1: Items 1-3. Start on Item 4
Week 2: Item 4 (possibly Item 5 also)
Week 3: (First half): Item 5 (if not already completed
Week 3: (Second half): Complete Item 5 and submit

Modifications and Errata: