Programming Design, Spring 2013
Homework 11

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

To submit your work, please upload the following one file to the online grading system PDOGS at http://stella.im.ntu.edu.tw/online-judgement/.

1. A CPP file for Problem 1 (to the PD012 section).

NO hard copy and NO late submission. The due time of this homework is ***1:00pm, June 3, 2013***.

Problem 1

(100 points) Please write a C++ program according to the following instructions.

What should your program do

In a small town, there were three rich ladies, Mary, Jane, and Alice. One day, one of them bought some new cars. Since then, all of them started to buy cars. You need to write a C++ program to keep tract on the status of these car owners and their cars. Below are some attributes that you need to record:

- For a car owner, attributes that you need to record include her name (a string with no space), the number of cars she owns (an integer) and a list of those cars. As the number of cars one owns may change from time to time, you should not use a fixed-size array to store these cars. Instead, pointers and dynamic memory allocation is required.
- For a car, attributes that you need to record include the plate number (a string), the mileage per litter (an integer in kilometer),[1] the total mileage (an integer in kilometer), the amount of remaining gasoline (an integer in litter), and whether the car is dirty (a Boolean variable).

One may buy a new car. While a new car will be given a plate number and its mileage per gallon, it must have no gasoline, zero mileage, and be clean. One may also dispose a car. The car will then be destroyed and disappear forever. Finally, one may sell a car to another car owner.

Regarding a single car, the following events may occur:

- One may drive a car by specifying the amount of gasoline she wants to use. If the amount she specifies is higher than the remaining amount, all gasoline will be consumed; otherwise, some gasoline should remain there. In either case, the mileage should increase based on the amount of gasoline used and the mileage per litter of this car.
- A car may become clean once the car is washed. After a car is washed, once the car travels more than 100 kilometers, it becomes dirty until it is washed again.
- A car can be refilled. The amount of remaining gasoline will increase according to the amount specified during the refilling process. For simplicity, assume that the gasoline tank has unlimited capacity.

Your program will process a list of events happened to this town. An event is recorded in one line. In each line, two pieces of information are separated with a white space. Below are some events and their formats:

[1] This means how many kilometers this car can travel for one litter of gasoline. For simplicity, we assume that this number is deterministic and fixed.

- A car owner buys a new car: The line contains a character `A`, a string of a car owner's name, a string of plate number, and a number representing the mileage per litter in order.
- A car owner disposes a new car: The line contains a character `D`, a string of a car owner's name, and a string of plate number in order.
- A car owner sells a car to another car owner: The line contains a character `S`, a string of the seller's name, a string of the buyer's name, and a string of plate number.
- A car owner drives one of her cars: The line contains a character `G`, a car owner's name, a car's plate number, and an amount of gasoline that the owner wants to consume. Note that the amount of remaining gasoline may not be enough.
- A car owner refills one of her cars: The line contains a character `R`, a car owner's name, a car's plate number, and an amount of gasoline added into the tank.
- A car owner washes one of her cars: The line contains a character `W`, a car owner's name, and a car's plate number. Note that a clean car may still be washed.

Below are some more events that each requires one line of output:

- A car owner asks for the current status of one of her car. When this happens, the line contains a character `I`, a car owner's name, and a plate number. Your program should output the car's plate number, mileage per litter, total mileage, amount of remaining gasoline, and whether the car is clean (`1` for clean and `0` for dirty). Two attributes should be separated by one white space.
- A police officer asks one owner to list all her cars. When this happens, the line contains a character `P` and a car owner's name. The output should be all the plate numbers of the owner's cars, according to the order of the time she obtains the car. In other words, the first car she buys should be listed first, then the second one she buys, then the third, etc. Two plate numbers should be separated by one white space.
- A boy looks for all the cars that are dirty: This little boys wants to earn some money by washing cars for car owners. Therefore, he may want to find all the dirty cars. When this happens, the line contains only one character `L`. Your program should output the plate numbers of all dirty cars in the alphabetical order.[2] Two plate numbers should be separated by one white space. If no car is dirty, output an empty line.
- Two cars are compared to find the one with the higher mileage per litter. When this happens, the line contains a character `C`, the first car owner's name, the first car's plate number, the second car owner's name (may be identical to the first owner), and the second car's plate number. Your program should output the plate number of the car with the higher mileage per litter. If there is a tie, output `tie`.
- All the cars in this town are compared to find the one with the highest current mileage. When this happens, the line contains a single character `M`. Your program should output the winner's plate number.

You may assume that all the events given to you are valid. For example, there will be no record showing that Alice is checking the status of Mary's car, Stephanie (who does not exist in this story) buys a new car, or one compares an existing car with one that does not exist. All numbers recorded are integers.

Below is a set of example inputs and outputs, where outputs are provided in parentheses after a line of input that requires outputs.

[2] All the characters on a plate will be capital English characters.

```
A Alice AABBB 7
A Alice AACCC 8
A Jane DDFFF 6
R Alice AABBB 10
G Alice AABBB 6
I Alice AABBB              (AABBB 7 42 4 1)
I Alice AACCC              (AACCC 8 0 0 1)
A Alice AAAAA 9
P Alice                    (AABBB AACCC AAAAA)
C Alice AAAAA Jane DDFFF   (AAAAA)
R Jane DDFFF 50
G Jane DDFFF 60
I Jane DDFFF               (DDFFF 6 300 0 0)
M                          (DDFFF)
L                          (DDFFF)
D Jane DDFFF
M                          (AABBB)
```

Some suggestions

Below are some suggestions that may be helpful. You certainly do not need to follow all of them if you have other ways that work.

1. You are suggested to define two classes, one for car owners and one for cars.
2. When a car owner buys a new car, how to add this car to the current list of cars? Below we demonstrate one possible way. If there are n cars before this new car is bought, we should first dynamically allocate a space for $n + 1$ cars, then copy all the existing n cars into the new spaces, and finally store the new car in the $(n + 1)$th slot. The same idea applies to the case of disposing and selling a car.
3. As the number of cars one owns may change from time to time, you should not use a fixed-size array to store these cars. Instead, pointers and dynamic memory allocation is required.
4. In reading the input file, note that for all the possible events you know what will follow after the first character. Therefore, instead of reading the whole line into a very long string and then try to decompose the string into words, you may directly read words into appropriate string variables.

Grading criteria

Your program will be graded based on the following criteria:

- 70% of your grades for this program will be based on the correctness of your output. The online grading system will input a set of testing data, which includes many lines of events. Among all the events, 35 lines will require outputs. You may only see the grades of running your program on these data but cannot see the inputs and outputs. The 35 output lines count for 70 points, i.e., 2 points for each line.
- 30% of your grades for this program will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.