Programming Design, Spring 2016
Homework 13

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

Please upload one PDF file for Problem 1 and two CPP files for Problems 2 and 3 (optional) to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***2:00 am, June 13, 2016***. Please answer in either English or Chinese. The maximum point of this homework is 120.

Before you start, please read Chapters 12 and 13 of the textbook.[1]

The TA who generates the testing data and grades this homework is ***Parker Chiang***.

Problem 1

(20 points; 5 points each) Please answer the following questions regarding the class `string`.

(a) Consider the program on page 11 of the slides. Explain why there will be a compilation error. Modify the copy constructor to fix the problem.

(b) Consider the parent class `MyVector2D` and child class `NNVector2D` introduced in class. Suppose that there is a public member function `reflect` in `MyVector2D`

```
void MyVector2D::reflect()
{
  this->x = -1 * this->x;
  this->y = -1 * this->y;
}
```

which reflects a vector to its opposite position with respect to the origin. Obviously, this function should not be invoked by an `NNVector2D`. If you are given the definition of `MyVector2D`, you are not able to modify it, and you want to inherit `MyVector2D` to construct `NNVector2D`, what should you do?

(c) Consider the classes `Character`, `Warrior`, and `Wizard` introduced in class. Now in this game, a character may eat a magical medicine to advance to her/his next level. In this case, her/his experience points will become the lowest points needed for being at that level. Declare a virtual public member function

```
void Character::magicalLevelUp() = 0;
```

for this and then implement this function in the two child classes to override this function.

(d) Consider the second program on page 48 of the slides. Explain why there is a compilation error at

```
Character c[3];
```

Problem 2

(55 points) Continue from Homework 12, let's refactor our program by using inheritance.

[1] The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.

Obviously, both passengers and drivers are "users" of the platform. We may also see that our definitions of the two classes `Passenger` and `Driver` share some common components. Therefore, it would be natural to create a class `User`, and then implement `Passenger` and `Driver` as its derived classes. The class `User` would contain the following member variables:

```
class User
{
protected:
  static int offlineCnt;
  static int searchingCnt;
  static int travelingCnt;
  int id;
  int status;
  int x;
  int y;
  int ratingCnt;
  int* ratings;
};
```

Note that the visibility modifier `private` has been changed to `protected`. Also note that not the three static variables refer to the number of offline, searching, and traveling "users." Later when your `Passenger` and `Driver` inherit `User`, it is suggested for you to add three new static variables instead of override these three static variables.

In this problem, all you need to do is to revise your program for Problem 2 of Homework 12. The calculations that need to be carried by your program are exactly the same as those in Problem 2 of Homework 12. All we want is to see your program get revised with inheritance.

Input/output formats

There are 15 input files. The input/output formats of the testing data will be the same as those for Problem 2 of Homework 12.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use only techniques covered so far. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

You must use inheritance to write your program. If you fail to do so, you will get no point. If you do, you will be graded according to the following rule:

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 25 points 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.

Problem 3

(45 points) This problem is a complication of Problem 2. Now a passenger may pay some money to become a VIP passenger. Those who are not VIP passengers are usual passengers. When a VIP

passenger searches for a car, she will be provided the list of all available drivers, and she may pick anyone she likes. A VIP passenger is never restricted by her average rating. All other features of this problem is the same as Problem 2.

Input/output formats

There are 15 input files. Below are some modifications from Problem 2:

1. (Event P) After the letter P, there is first an integer 0 or 1 for the passenger level and then an integer as the passenger ID. A VIP passenger's level is 1 while a usual passenger's level is 0.
2. (Event S) After the letter S, there is an integer as the passenger ID, an integer as x, and then an integer as y, where (x, y) is the passenger's current location. If the passenger is a VIP passenger, we then have an integer as the driver ID that is picked by the VIP passenger, if there is at least one available driver. In this case, that ID will be one of the IDs of the available drivers. If there is no available driver, the input line stops at y. j

For example, for the input

```
4
P 0 1
P 1 2
D 1
D 4
D 3
O 1 0 0
S 1 10 10
S 2 5 5
A 1 300 300 5 2
This driver is awesome!
He was LATEEEEEEE!!!!
C 1
O 1 10 11
O 4 10 10
O 3 6 6
S 2 10 9 1
S 1 10 10
D 2
O 2 -1 -1
C 3
```

Please focus on four lines:

- `P 0 1`: This means that passenger 1 is a usual passenger.
- `P 1 2`: This means that passenger 2 is a VIP passenger.
- `S 2 5 5`: As there is no available driver when passenger 2 searches for a car for the first time, no driver ID is given at the end.
- `S 2 10 9 1`: As there is at least one available driver when passenger 2 searches for a car for the first time, a driver ID is given at the end. Here it is indicated that driver 1 is picked by passenger 2.

It then follows that the output of this problem is

```
4 2
```

which are the IDs of available drivers at the end, ordered by their registration times.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use any technique. You are not even required to use the classes defined in Problem 2. Finally, you should write relevant comments for your codes.

Grading criteria

45 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 3 points.