Assignment 2

The American University in Cairo

Computer Science and Engineering Department

Fundamentals of Computing II

CSCE110101/02

Object Oriented Programming fundamentals

In this assignment, you will implement a car workshop appointment system, where customers can book appointments. The program should be able to take a customer's information and the appointment that they'd like to book and assigns them to a mechanic according to their availability. It should also be able to display the customers' names in order of their appointments (ascendingly).

Classes inheritance and overloading

First, Implement the following classes as explained below.

1. **Class Person**: An abstract parent class that has the following:
    - **Member variables:**
        - Name
        - ID
        - Age
    - **Methods(member functions):**
        - A default constructor setters getters *printInfo*: a print function
        - that prints the person's information.
        -
        -

- **Struct Appointment**: struct that has the following member variables:
    - hours
    - mins.

    Note: You can assume we're using the 24 hours system)

2. **Class customer**: child class inherits from class Person with the following:

- **Private variables:**
  - *MechanicID*: to indicate the ID of the mechanic assigned to their case. (can be any number)
  - *appointment*: struct of the type defined in the previous step to indicate the customer's appointment with the mechanic.
- **Member functions:**
  - setters
  - getters
  - Overload the <,>,== operators to compare between customer's appointments.

Example: two customers are equal if their appointments have the same values for hours and minutes.

3. **Class mechanic**: inherits from Person with the following:
   - **Private variables :**
     - *counter* : number of appointments the mechanic has for the day
     - Array of struct (appointment): that indicates the times the mechanic is booked.
   - **Methods:**
     - *isAvailable* A function that returns true if the mechanic is available at a certain time, false if not.
     - setters
     - getters

Template Class Queue

Second, implement a generic queue class to work with any data type. The class should at least support the following functions:

- Push
- Pop

Workshop Appointment program

Finally, combine the three classes(Person, Customer, mechanic) and Queue class to simulate a workshop appointment system. The workshop system works as follows:

1. The workshop has a number of mechanics available.
2. The customers come and choose an appointment time.
3. The system chooses the first mechanic available in the appointment time chosen by the customer and assigns this mechanic to the customer.
4. We assume that a mechanic is available 24 hours each day until a customer reserves one-time slot.

- **i.e.** A mechanic is not available at a certain time only if a customer already reserved this time, otherwise assume the mechanic is available.

5. If there are no available mechanics in the time chosen by the customer, then the customer will appointment will be canceled.
   - **i.e.** when printing the customers, print a message indicating that no mechanics were found for this customer.

Program Workflow :

1. You could read multiple mechanics' data from the user or initialize them within the main function. (use cin)
2. Read multiple customers' data from the user. (use cin)
   - Insert customers with their appointment into an array of customers.
3. The program should assign each customer to the mechanic in the order the mechanics are stored so the first customer would go to the first mechanic, the second customer to the second mechanic, and so on till you go back to the first mechanic.
4. If a mechanic is not available in the appointment the customer chose, then the next mechanic should be considered, and so on.
   - **Tip:** Use the mechanic class functions to check the mechanic's availability.
5. The customers should be inserted into a queue in order of their appointments.
   - For example, the customer with an earlier appointment is stored at the beginning of the queue and the customer with the latest appointment is stored at the end.
   - **Tip:** use the overloaded operators for the customer class to compare between them.
6. Print the information of the customers in order, along with the info of their assigned mechanic.

Test Case:

Input:

- **customer**:
  Ahmed at 1:00, Sara at 4:00, Kareem at 3:00, Mohammed at 1:00
- **mechanics** :
  Ayman, Khaled, Mai

Output:

Mr. Ahmed should be assigned to Ayman at 1:00

Mrs. Sara should be assigned to Khaled at 4:00

Mr. Kareem should be assigned to Mai at 3:00

Mr. Mohammed should be assigned to Khaled at 1:00 (since Ayman is unavailable at 1:00 and Khaled is next in the array)

After inserting the customers into the queue, they should be in the following order:

Mr. Ahmed has an appointment at 1:00. with Ayman

Mr. Mohammed has an appointment at 1:00 with Khaled

Mr. Kareem has an appointment at 3:00 with Mai

Mrs. Sara has an appointment at: 4 : 0. with Khaled

Bonus: Use files for the input.

Store the input for the customers/mechanics in a file and allow your program to read the input from the file directly.

We read the file line by line and save the data in the corresponding variables in the program. So each time we run the program the array of customers/mechanics can be already filled with data (In which case we don't take them from the user) **Example for the mechanics.txt:**

```
Mai // mechanic name
40 // mechanic age
1 // mechanic ID
 3 // number of appointments, should be followed by the number of
appointments*2 lines that indicate the hour & mins of each appointment.
   1 // appointment 1 at 1:00
   0
   5 // appointment 1 at 5:30
   30
   3 // appointment 3 at 3:15
   15
```

Then we read for each mechanic in the array of mechanics :

```
  mechanic x;
x.name= ((The first line which is the mechanic name))
  x.age= ((The second line which is mechanic age))
    x.ID= ((The third line which is mechanic ID ))
```

Then loop according to the number of appointments(given in line 4) to read the time of each appointment line by line into the x.appointments array. So the hours for appointment 1, for example, would be stored in x.appointments[0].hours, and the minutes for appointment 1 would be stored in x.appointments[0].mins.

Tips: to read customers/mechanics from a file easily, make sure to use the same format for each mechanic/customer in the text file.

Guidelines & Resources

1. Divide your classes in separate files, each class should have a header file and a .cpp file to keep your code more organised and readable.

2. Please make sure to document your code by adding comments that explain your logic and the functions you're creating whenever possible. Preferrably the comments for classes would be in the header file(*if exists*).
3. For each class:
    - it is important to implement setters/getters for the private member variables.
    - Don't make a member variable public unless necessary.
4. Some resources that might help you:
    - cpp overloading
    - syntax templates
    - syntax files
    - read/write.

Deliverables

Upload all files + **a report that has multiple screenshots explaining your output** to GitHub and submit the link.

Grade Distribution:

- 70% code correctness.

- 30% Report.

By submitting this assignment, I affirm that I have followed AUC's Code of Academic Ethics and the work submitted is my own. I have not consulted unauthorized resources or materials nor collaborated with other individuals unless allowed.

Good Luck!