**Assignment 3 report**

By: Malak Mekky

ID: 900222132

**Text

Description automatically generatedText

Description automatically generatedOutput:**

The program outputs the problem correctly. Each mechanic can have multiple customers as long as they don’t have a collision in timings. For example, the reason ayman does not have the customer mohammed is because he has a clash with another customer at time 1:00.

**File handling**

I was not able to implement the bonus of reading from the files since I kept on getting errors. However, you can see that the information on the files are correct.

Calendar

Description automatically generated with low confidence

A picture containing table

Description automatically generated

**Explanation of each file**

Customer.h file:

// This file defines a class called Customer which is inherited from the base class of person. The Customer class has two private data members, an integer ID and an Appointment object. It also has public member functions including setters and getters, and overloaded comparison operators. The default constructor initializes the ID to 0 and the Appointment to 0 hours and 0 minutes.

Other than the setter and getter functions, there is a function which returns if the mechanic is busy and calls the appointment hours from the user input and compares it to the times the mechanic is busy. If they are, it returns false.

Mechanic.h file:

The Mechanic class has two private data members, an integer counter and an array of Appointment objects, and several public member functions. The default constructor initializes the counter to 0. The isAvailable() function checks if the mechanic is available at a certain time by iterating through the appointments array and comparing the given time to each scheduled appointment. If there is a match, it returns false, indicating that the mechanic is not available at that time. Otherwise, it returns true. The setAppointment() function sets an appointment for the mechanic at a given time by adding the appointment to the next available index in the appointments array and incrementing the counter. The getCounter() function returns the number of appointments currently scheduled for the mechanic. The getAppointment() function returns the appointment object at a given index in the appointments array.

Person.h file

This is the parent class of Person. The Person class has three protected data members, a string for the person's name, an integer for their ID, and an integer for their age, and several public member functions. The default constructor initializes all three members to zero or empty strings. The setName(), setId(), and setAge() functions set the name, ID, and age of the Person object, respectively. The getName(), getId(), and getAge() functions return the name, ID, and age of the Person object, respectively. The printInfo() function prints the name, ID, and age of the Person object to the console. Overall, the Person class provides a simple framework for creating objects that have a name, ID, and age.

The Appointment struct has two public data members, integers for hours and minutes. It also has two public member functions, getHrs() and getMins(), that return the hours and minutes, respectively, of the appointment. The Appointment struct provides a simple way to represent an appointment time, where the hours and minutes can be accessed and manipulated.

Queue.h file

This is a Queue class which implements a basic queue data structure. It has a dynamically allocated array for holding the data elements, as well as front and rear pointers for keeping track of the beginning and end of the queue. The class provides methods for pushing and popping elements, checking if the queue is empty, and accessing the front and rear elements. If an attempt is made to push an element onto a full queue, or pop an element from an empty queue, the class will print an error message.

Workshop.cpp file

The program includes all of the above header files.

The program asks the user for the number of mechanics and customers. It then prompts the user to enter the details of each mechanic, including their name, age, and ID. The details of each mechanic are written to a file called "mechanics.txt".

The should then enter the details of each customer, including their name and appointment time. The details of each customer are written to a file called "customers.txt".

Then a queue of customers is created. It then goes through each customer in the queue and assigns them to an available mechanic. If no mechanics are available, the program prints a message saying that no mechanics are available for that customer. If a mechanic is available, the program assigns the customer to that mechanic and prints the details of the mechanic and the customer's appointment time.