Project

Ride and Transportation Management system

Submission Dead Line: Monday 22/6/2021 till labs are open

— PROVIDE PROPER INDENTATION AND COMMENTS WITH YOUR CODE
— YOU MUST DEALLOCATE ALL MEMORY PROPERLY, YOUR CODE SHOULD NOT HAVE ANY **MEMORY LEAKS** OR **DANGLING POINTERS**.

Question: In this project you are going to design a small application for the ride and transportation management system for a vehicle rental company. Where users can book the rides or use goods transportation services provided by the company.

A) Vehicles

The company has three major categories of vehicles, first one is used for rides and second for transportation of goods, but some vehicles can also be used for both services ride and goods transportation, such as Bikes, Rikshaws.

Ride vehicles can be further categorized as small vehicles (Bike, Scooter, Car, Rickshaws), which can be used for rides within same city and large vehicles (Van, Bus, APV, Campervans, Caravan), which can be booked for long distance traveling.

Transportation vehicles are also divided in two sub categories light vehicles (Shahzor, Mazda-Titan) that can only be used for intercity goods transportation and heavy vehicles (Trucks, Tractor-Trailor, Car-Transporter, Tanker-Truck) which are used for intracity goods transportations.

B) Drivers

According to on the vehicles there are three types of drivers some have training and license for small vehicles only; second type has training and license for heavy vehicles only and third type involves drivers who have license of both types.

https://en.wikipedia.org/wiki/Driving_licence_in_Pakistan

In Pakistan, there are different categories of driving license.

1. **Motorcar/jeep:** motorcar/jeep driving license is valid for non-commercial car.
2. **Motorbike/rickshaw**
3. **LTV**: light transport vehicle driving license is valid for commercial car/taxi, jeep, mini bus and lightweight transport.
4. **HTV**: Heavy transport vehicle driving license is valid for buses, trucks, trailers, cranes, and any type of heavy transport.
5. **Tractor** (agricultural)
6. **PSV**: public service vehicle
7. **International driver's permit**

C) Customers:

Customers can book rides or use the goods transportation services of company.

The record of all customers and drivers will be maintained in this system. If a customer books a ride or use transport service first time, then his/her details will be added in the system and then booking will be made on the basis of service the customer wants to use. If the record of a customer already exists then booking will be made on previous ID. If a customer completes a ride or his goods are transported to destination, then status of ride or delivery should be changed to complete and customer's history should be updated by adding new record to it. Similarly, driver's rides or transported orders list should be updated too.

A Customer can rank (at 0 to 5 scale worst to best respectively) both the vehicle, and driver, and this ranking will increase the overall ranking status of driver and a bonus of 15 % of salary should be given to all drivers with overall ranking status above 4.5. Similarly, the vehicle ranking will prioritize the vehicle for more ride services, and customer can select vehicles based on ranking from the available vehicles list.

The fair calculation for each service will be affected by the distance traveled, fuel type and cost, ranking and experience of the driver and also by ranking and additional features of vehicle, such as air conditioning, mileage and comfort level. The goods transportation services fair will be calculated by size of vehicle and distance covered. The skeleton of classes is provided below for this system, complete the implementation of all classes.

```
class Name{
private:
	char * fName;
	char * lName;
public:
	Name(char* fN, char* lN);
	//add following functions //Getter, Setters
	//Destructor, Default Constructor, Copy Constructor,
	friend ostream & operator<<(ostream & out, const Name & n);
};

ostream & operator<<(ostream & out, const Name & n) {
	out << n.fName << " " << n.lName << endl;
	return out;
}

class Date(){
	int day;
	int month;
	int year;
	//add constructurs getter setters and stream output function, which output date in
given format day / Mon / Year

};
class mTime(){
	int hour;
	int min;
	int sec;
	//add constructurs getter setters and stream output function, which output time in
given format Hr : Min : Sec};
```

```
class Service{
	char * source;
	char * destination;
	float  distance; //in km
	Date   bookingDate;
	mTime  bookingTime;
	bool   status; // false for pending, true if complete
	float  fuelrate;

	int    cId; // Customer Id who booked the ride
	int    dId; // Driver Id
	int    vId; // vehicle Id

	//you can add more members here if required
	//add member functions

};

class Ride: public Service{
	int    noofPassengers; // false for pending, true if complete
	char * rideType;       // intercity, intracity
	float  DriverRanking;  // 0 to 5 scale (worst to best)
	float  VehicleRanking; // 0 to 5 scale (worst to best)

	//you can add more members here if required
	//add member functions


};

class Delivery: public Service{
	float  goodsWeight; // Weight of goods in Kg
	char * goodsType;   //type of goods food, furniture, petroleum, chemicals, etc.
	//you can add more members here if required
	//add member functions
};
```

```
class Person {
private:
	Name pName;
	Date DOB;
	int Age;
	int Nid;
public:
	//add following functions in this class //Getter, Setters
	//Destructor, Default and Parametrized Constructor, Copy Constructor, Output
operator (print information of person)
};

class Customer: public Person{
private:
	int cId;
	// Unique and assigned first time when customer makes first service request
	Service ** bookingHistory;
	//Maintain and Update history of customer for each service (ride or goods
	transportation order) by adding address of service in dynamic array.
	//you can add more members here if required
public:
	//add following functions in this class //Getter, Setters
	//Destructor, Default and Parametrized Constructor, Copy Constructor, Output
operator (print complete information of customer including history if any)
};

class Driver: public Person{
private:
	int  dId;
	char ** LicencesList;
	int noofLicences;
	float overallRanking;
	float salary;
	int experience;
	int  status; // 1 for free, 2 if booked and 3 if canceled
	Service ** serviceHistory;
	//Add services address in the array for tracking complete information of service.
public:
	//add following functions in this class
	//Getter, Setters
	//Destructor, Default and Parameterized Constructor, Copy Constructor, Output
operator (prints complete information of Driver)

};
```

```
class Feature{

	int  fId;
	char * description;
	float  cost;
	//you can add more members here if required
	//add member functions
};
```

```
class Vehicle{

       int  vId;
       int  registrationNo;
       float avgMileage;
       char * LicenceType;  // License required for driving the vehicle
       bool  status; // false for free, true if booked in a service already
       char * fueltype;
       float overallRanking;
       Date  manufacturingDate;
       Feature * list;
       Service ** serviceHistory;
       //Add services address in the array for tracking complete information of service.

       //you can add more members here if required
       //add member functions
};
//add Complete Hierarchy of vehicles based on their types and functions
```

D) Using classes designed above, write a program (**TMS class**) to simulate the Transport Management System. You will need to create dynamic arrays of Drivers, Customers, Vehicles and Services (ride, delivery). Write a program that can process up to 15 Customers, 10 Drivers, and 20 vehicles. Your program should contain a menu that should provide the user different choices to effectively run the program; in other words, your program should be user driven with at least following functionalities.

1. Add a new Customer,
2. Add or remove a Driver
3. Add or remove a Vehicle
4. Print List of All Customers
5. Print List of All drivers
6. Print complete list of vehicles with details by their category.
7. Print complete details and service history of a particular vehicle, (Provide vehicle ID)
8. Print complete history of a particular customer, (Provide customer ID)
9. Print complete history of a driver including services. (Provide driver ID)
10. Print list of all drivers who have ranking above 4.5.
11. Print list of all drivers who have multiple licenses and their license information too.
12. Print updated salary of all drivers based on their updated ranking.
13. Add a Service request (ride or delivery) for a customer. Customer will be provided with a list of free vehicles and drivers to choose based on their ranking in descending order. The fair for the service should be calculated and displayed to the user.
14. When the customer cancels a booking, he will be charged 5% of service fair cost on cancelation. (Provide service and customer ID). The driver and vehicle status should be

updated after cancelation and no record of service should be added in driver and vehicle, but customer bookings list should be updated.

15. Complete a service (Provide service ID). Update Customer, Driver's, and vehicles service record once a service status is completed, and keep that service in Services list too. Customer should add ranking of a driver and vehicle for a service after completion.
16. Print details of all Customers, who used the same vehicle in a ride service on a different date.
17. Print the List of all Drivers who completed delivery services on same days.
18. Print details of all pending services on a particular date.
19. Print details of all pending services of a particular driver.
20. Print details of all canceled services by a customer.

E) Write a main program, which will test all functions of **TMS** class. You should read and update Services, Vehicles, Drivers and Customers information by creating separate files as database. When application will start the data of all objects should be loaded from files in the lists, and on termination you will store updated data of lists in files.

Note: You can take assumptions where needed but use proper relationships in classes.