MIDTERM EXAMINATION

Course: **CSC10003 OBJECT-ORIENTED PROGRRAMMING**
Time: **90** minutes (**100 points**) Term: 1 – Academic year: **2024-2025**
Lecturer(s): **Le Khanh Duy**

(Notes: Closed-book exam)

1. (**10 points**) What is encapsulation and data hiding? Why do we have to hide the data? Give an example to illustrate.
2. (**10 points**) What are constructors and destructors used for? Give examples.
3. (**30 points**) Add your source code to make the following program works correctly:

```
#include <iostream>
using namespace std;

//Assuming that you ALREADY have the class Book fully implemented
//with all needed functions/methods
class BookStore {
public:
    //add your code here
private:
    Book* arrBook; //a dynamic array to store books
    int n;
};

int main() {
    //a store with 100 slots with "default books"
  //assume that Book has all methods pre-built and ready
    BookStore storeA(100);

    //ID, Title, Price, Year. Don't need to build this method for Book!
    Book x(1234, "English", 7.9, 2003);
    Book y(5678, "Computer Science", 12.5, 2020);

    storeA.put2Slot(0, x); //put the book x into slot 0
    storeA.put2Slot(1, y); //put the book y into slot 1
    BookStore storeB(10);
    storeB = storeA; //storeB is the same as storeA

    //print out each book from n slots in the store
    cout << "My store: " << storeB << endl;

    return 0;
}
```

4. **(50 points)** You are tasked to develop a management system for a vehicle rental store. The store provides different types of vehicles for rent including Cars, Trucks and Bikes. All vehicles should have the following information: a ***make*** (string) representing the manufacturer (e.g., Toyota), a ***model*** (string) representing the model of the vehicle (e.g., Corolla Cross), a ***license plate*** (string) to uniquely identify the vehicle, and a ***daily rate*** specifying the rental rate per day.

   Besides that, each type of vehicle also has its also specific attributes. A car contains the information of seating capacity indicating the number of seats. A truck has cargo capacity representing its cargo carriage capacity in tons. A bike has an attribute which indicates if renters are required to wear helmets when riding. Overall, a vehicle has two ***rent*** methods.

   - rent(): only prints out that the vehicle has been rented and records the current date as the starting day of the rental.
   - rent(customer_id): where customer_id is a string representing the ID of the customer. This method prints out that the vehicle has been rented by the customer with customer_id, and records the current date as the starting day of the rent.

   Each vehicle also has a ***return*** method which will record the current date as the ending date of the rent.

   Each vehicle also has a ***get_rent_cost***(start_day, end_day) method to calculate the total cost for a rental period, where ***start_day*** and ***end_day*** are two Date input arguments, indicating the starting and ending day of the rental period.

   Note: we can assume that class Date and all necessary attributes and methods (e.g., subtraction of two Date objects, get current date, etc.) already exist.

   The pricing options of the vehicle types are as follow:

   - Car:
     - Daily rate: $40 per day.

  - ***get_rent_cost*** (start_day, end_day) method returns the estimated rental cost calculated based on the daily rate and the rental period.
- Truck:
  - Daily rate: $70 per day.
  - Adds a surcharge of $10 per ton of cargo capacity per day to the base rate.
  - ***get_rent_cost*** (start_day, end_day) method returns the estimated rental cost calculated based on the daily rate, the rental period and the cargo capacity surcharge.
- Bike:
  - Daily rate: $10 per day.
  - If the renter is required to wear a helmet, a $5 helmet rental fee is added.
  - ***get_rent_cost*** (start_day, end_day) method returns the estimated rental cost calculated based on the daily rate, the rental period and the helmet rental fee (if applied).

The rental system manages a collection of vehicle objects. The system should provide the following functions:

- **add_vehicle**: Add new vehicles to the collection. This method needs to check if the license_plate of the vehicle already exists in the collection or not. If the license_plate already exists, the vehicle cannot be added to the collection.
- **rent_vehicle_to_customer**: Accept a license_plate and a customer_id, check if the vehicle with the license_plate is still available for rent. If yes, print out the vehicle with the license_plate has been rented to the customer with customer_id and mark the vehicle as rented.
- **return_vehicle:** Accept a license_plate. This method looks for the vehicle with the given license_plate, performs the return of the vehicle, prints out the total rental costs and marks the vehicle as available for rent.

a) **(20 points)** Please draw a UML diagram of all possible classes in the problem above. The representation of each class in the diagram should provide sufficient details such as attributes and their data types, methods including return data types and well names and datatypes of input argument, whether members are static (<<static>>) or whether methods are virtual (<<virtual>>), etc.
b) **(20 points)** Write code to implement methods of the classes identified above.
c) **(5 points)** Write a test function to demonstrate how you use the declared classes to manage an exemplary store.
d) **(5 points)** Perform necessary modifications and additions in the code to calculate the total rental costs of the store from a starting to an ending day.

---END---