Object Oriented programming

Assignment 2

Due Date: 6 August 2022.

Total marks 130

CLO: 5

Note: Do not copy code from internet or any other source even across the sections. I will check the plagiarism of the assignment and it will capture the cheated code from internet or copied from class fellows. Those who will try they will get some marks for their effort even you do not get perfect solution. Copied & Shared work will score in negative grading. Assignment should be in visual Studio. You will send .c and .exe file in zipped format. Write "System Pause" at the end so that exe screen does not vanish instantly. After submission, no excuse will be entertained. No assignment will be accepted after due date.

Marks distribution
Q1: 4 parts →20 marks **main etc→10 marks**
Q3: 40 marks

Q1. A bus is a public transport road vehicle designed to carry significantly more passengers than the average cars or vans. Write a class called **Bus** that keeps trail of its mileage (number of miles it's driven), the number of gas left in its tank in gallons and Bus status: broken or not. You are required to create a class Bus and give an initial values to gas in the tank(decimal), and mileage(decimal):
Bus *m_bus = new Bus(20.0, 1200.0);
Bus class has a method StartAndDrive that has a parameter miles the bus should drive. The functionalities of this method are:

I. Whenever a bus starts for a ride, there is always a probability that it's going to break down. Precisely, for 0 to 10,000 miles, the probability of a failure is 0.1 (10%) and from 10,001 to 20,000 probability is 0.2 (20%) etc. If the bus goes out of order, it will stop driving.
II. On the other hand, if the bus starts successfully, it needs to cover milage which was given in parameter. Mileage and the remaining amount of gas should be updated accordingly to specify the bus has driven additional miles, and has used up some gas. A constant should be added to calculate MILES_PER_GALLON.
III. If the bus does not have enough fuel that it can cover the given distance then the bus should run out of gas. If bus was able cover the given distance, then function should return true, and false otherwise (if it goes out of order or runs out of gas).
IV. Write a method called testBus that makes a new Bus and returns the mileage of that bus once it breaks down for the first time. The bus should start out with 10 gallons of gas, and 0 miles driven, and you should drive the bus in increments of 10 miles. If the bus runs out of gas, you should refill it to 10 gallons of gas. Once the bus breaks down, you should return how many miles the bus had driven.

Example 1:

- m_bus->StartAndDrive (25).

- Answer will be true and bus has 9 gallons of gas and driven miles are 1025.
- MILES_PER_GALLON= 25
- Suppose, bus did not break down.

Example 2:

- m_bus->StartAndDrive (250).
- return false because of shortage of gas to cover this distance
- Bus have got 0 gallons of gas with 1,250 miles covered distance as to cover 225 miles which were left uncovered needs 9 gallons of gas.

Class methods:

```
bool StartAndDrive (float miles);
int getMileage();
bool statusofBus();  /* broken down or not
public void repair(); //Optional
void addFuel(double no_of_gallons);
```

Q2. Part-Whole (Has-a) relationships, specifically focusing on Composition and Association concepts and implementations, including bi-directional associations with multiplicity

Instructions:

- **Solve the following tasks exactly in the given order.**
- **Make a separate project for each task.**
- **Indent your code properly.**
- **Use meaningful variable and function names. Follow the naming conventions.**
- **Use meaningful prompt lines and labels for all input/output.**
- **Make sure that there are NO dangling pointers or memory leaks in your program.**

Task 1: 20+(4x10)

We have 4 classes: **MyString, Gun, Inventory and Player.**

MyString Class: data members are given in diagram.

Gun Class: represents a single gun. Initially all the guns are fully loaded with bullets. Once a gun is out of bullets you can't reload the bullets or shoot more fire.

Inventory Class: All the guns are created in the Inventory. Player will pick his/her guns from inventory. Once a gun is picked by some player, it will be unavailable for other players.

Player Class: Player can hold at max 3 guns at a time. Player should be able to fire the guns, put his/her gun back in inventory and pick new guns from inventory.

MyString

-cstr: **char***
-size: **int**

…

Gun

-ID: **int //unique**

-title: **MyString**

-bulletsCapacity: **int**

-noOfBullets: **int**

-fireSpeed: **float**

-damagePerBullet: **int**

+Gun(ID:**int**, title:**MyString**, bulletsCapacity:**int**, fireSpeed:**float**, damagePerBullet:**int**)

+displayInfo(): **void**

+reload(): **void**

+fire(): **void**

…

Player

-userName: **MyString** // unique

-health: **int**

Player can hold upto 3 guns. Which data member will you use to hold these guns?

-activeGun: **int** // 1,2 or 3. This integer represents which gun is active.

+fireTarget(enemy:**Player**): **void**

+fireTarget(): **void //Random shoots. no focused target.**

+displayInfo(): **void**

+receiveDamage(damage: **int**): **void**

+changeActiveGun(**?**): **void**

+exchangeGun(**?**): **void** // replace a gun from inventory.

A player should die when its health reaches to 0. Call destructor.

…

Inventory

-gun[100]: **Gun***

-gunStatus[100]: **bool*** //is gun assigned to player or not? A gun should be assigned to only one player.

-gunCount: **int**

…

In your driver class create 2 players and perform the following actions:

Task 1.1

Display the complete inventory info (how many guns are there in inventory and information of those guns. Inventory can have at max 100 guns.

Task 1.2

Assign 3 guns to both players and display the complete info of both players including their guns'
info.

Task 1.3

Make player 1 to shoot 6 fires on player 2 (2 fires from each gun player 1 have). Make player 2 to shoot 3 fires on player 1 (1 fires from each gun player 2 have). Display both players' info after fire exchange.

Task 1.4

Make player 1 shoot player 2 until he died. Now try to display both players'.

--

Q3. In the human history cryptography has played a vital role during wars, by encrypting the war messages before sending them and then decrypting them on the receiving end.

The simplest of encryption can be done by shifting the letters of the messages by "n" (where n can be an integer between 1 and 25). Alphabets forward, **for instance "Hello" can be encrypted as "Ifmmp" by shifting its each letter to the right by one Alphabet.**

In the context of this Assignment you will be given a space separated string encrypted with **penta-shifting encryption scheme**, means that each letter will be shifted 5 letters forward.

An important thing to note is that if a letter such as 'Z' is shifted 5 letters forward it will take a turn around and be shifted as Follows

Z->A->B->C->D->E

You are a software engineer working for a Tech Company and have been assigned with a task to design a Decoder to decrypt the message encrypted by penta-shifting scheme.

Once you have designed the decoder, your company gives you description of two advance types of decoders with additional features.

1) Given Below is the Description of what decoder of each type offers

1) **Decoder**

- Simply decodes the message and displays the output on the screen

2) **Super Decoder:**
   - Decodes the message and displays the output on the screen
   - Counts the number of words in the string
   - Counts the number of vowels and non-vowels
3) **Premium Decoder:**
   - Offers all the feature of Super Decoder in addition to displaying the decoded string in reverse order

You have been provided with a general structure of the classes to implement, it is now up to you to implement them correctly so they function as required.

```
//Note: You can add additional members to the class if required
class Decoder
{
private:
		string message;
public:
		Decoder(string)
		void Decode();
};

class SuperDecoder : public Decoder
{
public:
		SuperDecoder(string)
		void Decode();
};

class PremiumDecoder : public Decoder
{
public:
		PremiumDecoder(string)
		void Decode();
};
```

2) After you have designed the three decoders, your manager assigned you the task to design a single function that will accept a decoder object and offers the functionality of the type of Decoder Passed as the Parameter. To make your task easier he provides you with the following prototype:

```
void OperateDecoder(Decoder &Instance);
```