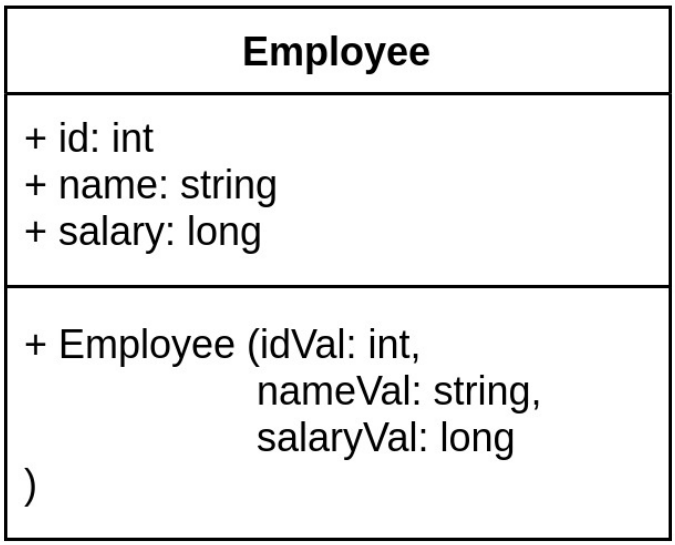
Lab Work 3

Question 1

Open Dev C++ and write the appropriate codes



UML Diagram for Employee Class

Do not modify the main method. Also use the same names and data types as used in the UML Diagram.

```
int main() {

    // your code goes here

    Employee hannibal_lecter(1, "Hannibal Lecter",
10000000);

    Employee norman_bates(2, "Norman Bates",
9000000);

    Employee darth_vader(3, "Darth Vader", 8000000);

    cout << hannibal_lecter.id << " " <<
hannibal_lecter.name << " " << hannibal_lecter.salary
<< endl;

    cout << norman_bates.id << " " <<
norman_bates.name << " " << norman_bates.salary <<
endl;
```

```
    cout << darth_vader.id << " " << darth_vader.name
<< " " << darth_vader.salary << endl;

    return 0;

}
```

Question 2 – inheritance

The Automobile, Car, Truck, and SUV classes

Suppose we are developing a program that a car dealership can use to manage its inventory of used cars. The dealership's inventory includes three types of automobiles: cars, pickup

trucks, and sport-utility vehicles (SUVs). Regardless of the type, the dealership keeps the following data about each automobile:

• Make

• Year model

• Mileage

• Price

Each type of vehicle that is kept in inventory has these general characteristics, plus its own

specialized characteristics. For cars, the dealership keeps the following additional data:

• Number of doors (2 or 4)

For pickup trucks, the dealership keeps the following additional data:

• Drive type (two-wheel drive or four-wheel drive)

And, for SUVs, the dealership keeps the following additional data:

• Passenger capacity

In designing this program, one approach would be to write the following three classes:

• A Car class with attributes for the make, year model, mileage, price, and number of doors.

• A Truck class with attributes for the make, year model, mileage, price, and drive type.

• An SUV class with attributes for the make, year model, mileage, price, and passenger capacity.

This would be an inefficient approach, however, because all three classes have a large number of common data attributes. As a result, the classes would contain a lot of duplicated

code. In addition, if we discover later that we need to add more common attributes,we would have to modify all three classes.

A better approach would be to write an Automobile base class to hold all the general data about an automobile, and then write derived classes for each specific type of automobile.