Inheritance (multiple & diamond)

- **Ex1**

```
#include <iostream>
#include <string>

using namespace std;

// Base class Vehicle
class Vehicle {
protected:
   string make;
   string model;

public:
   Vehicle(string make, string model) {
       this->make = make;
       this->model = model;
   }

   // Method to display make and model of the vehicle
   void displayInfo() {
       cout << "Make: " << make << endl;
       cout << "Model: " << model << endl;
   }
};

// Derived class Car inheriting from Vehicle
class Car : public Vehicle {
public:
   Car(string make, string model) : Vehicle(make, model) {}

   // Override displayInfo() method to display car info
   void displayInfo() {
       cout << "Car Information:" << endl;
       Vehicle::displayInfo(); // Calling base class method
   }
};

class Truck : public Vehicle {
public:
   Truck(string make, string model) : Vehicle(make, model) {}

   void displayInfo() {
```

```
        cout << "Truck Information:" << endl;
        Vehicle::displayInfo();
    }
};

int main() {
    // Creating instances of Car and Truck
    Car car("Toyota", "Camry");
    Truck truck("Ford", "F-150");

    // Displaying information using displayInfo() method
    car.displayInfo();
    cout << endl;
    truck.displayInfo();

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debu

Car Information:
Make: Toyota
Model: Camry

Truck Information:
Make: Ford
Model: F-150

D:\inheritance\x64\Debug\inheritance.exe (process 65
Press any key to close this window . . .
```

- **Ex 2:**

```
#include <iostream>
#include <string>

using namespace std;

// Base class Vehicle
```

```
class Vehicle {
protected:
   string make;
   string model;

public:
   Vehicle(string _make, string _model) : make(_make), model(_model) {}

   // Method to display make and model of the vehicle
   void displayInfo() {
       cout << "Make: " << make << endl;
       cout << "Model: " << model << endl;
   }
};

// Derived class Car inheriting from Vehicle
class Car : public Vehicle {
public:
   Car(string _make, string _model) : Vehicle(_make, _model) {}

   // Override displayInfo() method to display car info
   void displayInfo() {
       cout << "Car Information:" << endl;
       Vehicle::displayInfo(); // Calling base class method
   }
};


class Truck : public Vehicle {
public:
   Truck(string _make, string _model) : Vehicle(_make, _model) {}

   void displayInfo() {
       cout << "Truck Information:" << endl;
       Vehicle::displayInfo();
   }
};

// Derived class HybridCar inheriting from both Car and Truck
class HybridCar : public Car, public Truck {
public:
   HybridCar(string _make, string _model) : Car(_make, _model), Truck(_make,
_model) {}

   // Method to display hybrid car info
```

```
    void hybridInfo() {
        cout << "HybridCar: Make - " << Car::make << endl;
        cout<<", Model - " << Car::model << endl;
    }
};

int main() {
    // Creating instance of HybridCar
    HybridCar hybrid("Toyota", "Prius");

    // Demonstrating hybridInfo() method
    hybrid.hybridInfo();

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debu

HybridCar: Make - Toyota
, Model - Prius

D:\inheritance\x64\Debug\inheritance.exe (process 14424) exited with code 0.
Press any key to close this window . . .
```

- **Ex 3:**

```
#include <iostream>
#include <string>

using namespace std;

// Base class Employee
class Employee {
protected:
    string name;
    double salary;

public:
    Employee(string _name, double _salary) : name(_name), salary(_salary) {}

    // Method to display salary of the employee
    void displaySalary() {
        // This method will be overridden in derived classes
        cout << "Employee's salary is " << salary << endl;
```

```
    }
};

// Derived class Manager inheriting from Employee
class Manager : public Employee {
public:
    Manager(string _name, double _salary) : Employee(_name, _salary) {}

    // Override displaySalary() method to display manager's salary
    void displaySalary() {
        cout << "Manager's salary is " << salary << endl;
    }
};

// Derived class Engineer inheriting from Employee
class Engineer : public Employee {
public:
    Engineer(string _name, double _salary) : Employee(_name, _salary) {}

    // Override displaySalary() method to display engineer's salary
    void displaySalary() {
        cout << "Engineer's salary is " << salary << endl;
    }
};

int main() {
    // Creating instances of Manager and Engineer
    Manager manager("John Doe", 50000);
    Engineer engineer("Jane Smith", 60000);

    // Demonstrating displaySalary() methods
    manager.displaySalary();
    engineer.displaySalary();

    return 0;
}
```

Output:

```
Microsoft Visual Studio Debu
Manager's salary is 50000
Engineer's salary is 60000

D:\inheritance\x64\Debug\inheritance.exe (process 16328) exited with code
Press any key to close this window . . .
```

- **Ex4:**

```cpp
#include <iostream>
#include <string>

using namespace std;

// Base class Employee with virtual inheritance
class Employee {
protected:
   string name;
   double salary;

public:
   Employee(string _name, double _salary) : name(_name), salary(_salary) {}

   // Method to display salary of the employee
   void displaySalary() {
       // This method will be overridden in derived classes
       cout << "Employee's salary is " << salary << endl;
   }
};

// Derived class Manager inheriting from Employee
class Manager : virtual public Employee {
public:
   Manager(string _name, double _salary) : Employee(_name, _salary) {}

   // Override displaySalary() method to display manager's salary
   void displaySalary() {
       cout << "Manager's salary is " << salary << endl;
   }
};
```

```
class Engineer : virtual public Employee {
public:
   Engineer(string _name, double _salary) : Employee(_name, _salary) {}


   void displaySalary() {
       cout << "Engineer's salary is " << salary << endl;
   }
};

// Derived class TeamLead inheriting from both Manager and Engineer
class TeamLead : public Manager, public Engineer {
public:
   TeamLead(string _name, double _salary) : Employee(_name, _salary),
Manager(_name, _salary), Engineer(_name, _salary) {}

   // Method to display team lead information
   void teamInfo() {
       cout << "TeamLead: " << name << ", Salary: " << salary << endl;
   }
};

int main() {
   // Creating instance of TeamLead
   TeamLead teamLead("Alex Smith", 80000);

   // Demonstrating teamInfo() method
   teamLead.teamInfo();

   return 0;
}
```

Output:

```
Microsoft Visual Studio Debu

TeamLead: Alex Smith, Salary: 80000

D:\inheritance\x64\Debug\inheritance.exe (process 18544) exited
Press any key to close this window . . .
```

➢ **Ex5:**

```
#include <iostream>
#include <string>

using namespace std;

// Base class Device
class Device {
protected:
   string companyName;
   string size;
   string storage;
   string RAM;
   string model;

public:

   Device(string _companyName, string _size, string _storage, string _RAM, string
_model)
       : companyName(_companyName), size(_size), storage(_storage), RAM(_RAM),
model(_model) {}

   // Method to display device information
   void device_info() {
       cout << "\nDevice info: " << endl;
       cout << "Company Name: " << companyName << ", "<<endl;
       cout << "Size: " << size << ", "<<endl;
       cout << "Storage: " << storage << ", " << endl;
       cout << "RAM: " << RAM << ", "<<endl;
       cout << "Model: " << model << endl;
   }
};

// Derived class Phone inheriting from Device
class Phone : public Device {
public:
   // Constructor to initialize Phone specific attributes and call base class
constructor
   Phone(string _companyName, string _size, string _storage, string _RAM, string
_model)
       : Device(_companyName, _size, _storage, _RAM, _model) {}

   // Override device_info() method to display Phone device information
   void device_info() {
```

```
        cout << "Phone device info: ";
        Device::device_info(); // Call base class method to display common device
information
    }
};


class Laptop : public Device {
public:

    Laptop(string _companyName, string _size, string _storage, string _RAM, string
_model)
        : Device(_companyName, _size, _storage, _RAM, _model) {}


    void device_info() {
        cout << "Laptop device info: ";
        Device::device_info();
    }
};

int main() {
    // Creating instances of Phone and Laptop
    Phone phone("Apple", "5 inches", "256GB", "8GB", "iPhone 13");
    Laptop laptop("Dell", "15 inches", "512GB SSD", "16GB", "XPS 15");

    // Demonstrating device_info() methods
    phone.device_info();
    laptop.device_info();

    return 0;
}
```

Output:

```
Phone device info:
Device info: Company Name: Apple,
Size: 5 inches,
Storage: 256GB,
RAM: 8GB,
Model: iPhone 13
Laptop device info:
Device info: Company Name: Dell,
Size: 15 inches,
Storage: 512GB SSD,
RAM: 16GB,
Model: XPS 15
```

- **Ex6:**

```
#include <iostream>
#include <string>

using namespace std;

// Base class Device
class Device {
protected:
    string companyName;
    string size;
    string storage;
    string RAM;
    string model;

public:
    // Constructor to initialize attributes
    Device(string _companyName, string _size, string _storage, string _RAM,
string _model)
        : companyName(_companyName), size(_size), storage(_storage), RAM(_RAM),
model(_model) {}


    void device_info() {
        cout << "Device info: ";
```

```
        cout << "Company Name: " << companyName << ", \n";
        cout << "Size: " << size << ",\n ";
        cout << "Storage: " << storage << ",\n ";
        cout << "RAM: " << RAM << ", \n";
        cout << "Model: " << model << endl;
    }
};

// Derived class Phone inheriting from Device
class Phone : virtual public Device {
public:
    // Constructor to initialize Phone specific attributes and call base class
constructor
    Phone(string _companyName, string _size, string _storage, string _RAM, string
_model)
        : Device(_companyName, _size, _storage, _RAM, _model) {}

    // Override device_info() method to display Phone device information
    void device_info() {
        cout << "Phone device info: ";
        Device::device_info(); // Call base class method to display common device
information
    }
};


class Laptop : virtual public Device {
public:

    Laptop(string _companyName, string _size, string _storage, string _RAM,
string _model)
        : Device(_companyName, _size, _storage, _RAM, _model) {}


    void device_info() {
        cout << "Laptop device info: ";
        Device::device_info();
    }
};

// Derived class Tablet inheriting from both Phone and Laptop
class Tablet : public Phone, public Laptop {
public:
    // Constructor to initialize Tablet specific attributes and call base class
constructors
```

```
    Tablet(string _companyName, string _size, string _storage, string _RAM,
string _model)
        : Phone(_companyName, _size, _storage, _RAM, _model),
Laptop(_companyName, _size, _storage, _RAM, _model), Device(_companyName, _size,
_storage, _RAM, _model) {}

    // Method to display tablet information
    void device_info() {
        cout << "Tablet device info: ";
        Device::device_info();
    }
};

int main() {
    // Creating instance of Tablet
    Tablet tablet("Samsung", "10 inches", "256GB", "8GB", "Galaxy Tab S7");

    // Demonstrating device_info() method
    tablet.device_info();

    return 0;
}
```

Output:

```
Tablet device info: Device info: Company Name: Samsung,
Size: 10 inches,
 Storage: 256GB,
 RAM: 8GB,
Model: Galaxy Tab S7

D:\inheritance\x64\Debug\inheritance.exe (process 10596) exited with cod
Press any key to close this window . . .
```