Subject:- C++
LAB Assignment - 8

1.Write a program of single inheritance.

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

// Base class
class Animal {
protected:
    std::string name;

public:
    // Constructor
    Animal(std::string n) : name(n) {}

    // Method to return a generic message
    virtual std::string speak() {
        return "I am an animal.";
    }
};

// Derived class
class Dog : public Animal {
public:
    // Constructor
    Dog(std::string n) : Animal(n) {}

    // Overriding the speak method
```

```
    std::string speak() override {
        return "Woof! My name is " + name;
    }
};

int main() {
    // Create an instance of Dog
    Dog myDog("Buddy");

    // Call the speak method
    std::cout << myDog.speak() << std::endl;  // Output: Woof! My name
is Buddy.

    return 0;
}
```

output:

```
Enter Employee Data:
Enter Name: Alice
Enter Age: 30
Enter Employee ID: 4567

Employee Details:
Name: Alice
Age: 30
Employee ID: 4567
```

2.Write a program of multiple inheritance.

```
#include <iostream>
```

```
#include <string>

// Base class 1
class Animal {
public:
    void eat() {
        std::cout << "Animal is eating." << std::endl;
    }
};

// Base class 2
class Pet {
public:
    void play() {
        std::cout << "Pet is playing." << std::endl;
    }
};

// Derived class
class Dog : public Animal, public Pet {
public:
    void bark() {
        std::cout << "Woof! Woof!" << std::endl;
    }
};

int main() {
    // Create an instance of Dog
    Dog myDog;

    // Call methods from both base classes and the derived class
    myDog.eat();   // Output: Animal is eating.
```

```
    myDog.play();  // Output: Pet is playing.
    myDog.bark();  // Output: Woof! Woof!

    return 0;
}
```

output:

```
Enter Employee Data:
Enter Name: David
Enter Age: 29
Enter Organization Name: XYZ Tech
Enter Employee ID: 7890

Employee Details:
Name: David
Age: 29
Organization: XYZ Tech
Employee ID: 7890
```

3.Write a program of multilevel inheritance in c++.

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

// Base class 1
class Animal {
public:
    void eat() {
        std::cout << "Animal is eating." << std::endl;
```

```
    }
};

// Base class 2
class Pet {
public:
    void play() {
        std::cout << "Pet is playing." << std::endl;
    }
};

// Derived class
class Dog : public Animal, public Pet {
public:
    void bark() {
        std::cout << "Woof! Woof!" << std::endl;
    }
};

int main() {
    // Create an instance of Dog
    Dog myDog;

    // Call methods from both base classes and the derived class
    myDog.eat();   // Output: Animal is eating.
    myDog.play();  // Output: Pet is playing.
    myDog.bark();  // Output: Woof! Woof!

    return 0;
}
```

output:

```
Enter Manager Data:
Enter Name: John
Enter Age: 35
Enter Employee ID: 1001
Enter Department: Sales

Manager Details:
Name: John
Age: 35
Employee ID: 1001
Department: Sales
```

4.Write a program of Hybrid inheritance.

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

// Base class
class Person {
public:
    void getPersonData() {
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter Age: ";
        cin >> age;
    }
    void displayPersonData() {
        cout << "Name: " << name << endl;
```

```
        cout << "Age: " << age << endl;
    }

protected:
    string name;
    int age;
};

// Derived class from Person (Single Inheritance)
class Employee : public Person {
public:
    void getEmployeeData() {
        getPersonData();  // Getting data from the base class
        cout << "Enter Employee ID: ";
        cin >> empID;
    }
    void displayEmployeeData() {
        displayPersonData();  // Displaying base class data
        cout << "Employee ID: " << empID << endl;
    }

protected:
    int empID;
};

// Another base class
class Organization {
public:
    void getOrganizationData() {
        cout << "Enter Organization Name: ";
        cin >> orgName;
    }
```

```
    void displayOrganizationData() {
        cout << "Organization: " << orgName << endl;
    }

protected:
    string orgName;
};

// Derived class inheriting from both Employee and Organization
(Multiple Inheritance)
class Manager : public Employee, public Organization {
public:
    void getManagerData() {
        getEmployeeData();      // Getting data from Employee class
        getOrganizationData();  // Getting data from Organization class
        cout << "Enter Department: ";
        cin >> department;
    }
    void displayManagerData() {
        displayEmployeeData();      // Displaying Employee class data
        displayOrganizationData();  // Displaying Organization class data
        cout << "Department: " << department << endl;
    }

private:
    string department;
};

int main() {
    Manager mgr;

    cout << "Enter Manager Data:" << endl;
```

```
    mgr.getManagerData();

    cout << "\nManager Details:" << endl;
    mgr.displayManagerData();

    return 0;
}
```

output:

```
Enter Manager Data:
Enter Name: Sarah
Enter Age: 40
Enter Employee ID: 1234
Enter Organization Name: ABC Corp
Enter Department: HR

Manager Details:
Name: Sarah
Age: 40
Employee ID: 1234
Organization: ABC Corp
Department: HR
```

5.Write a program for Virtual Base Class.

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

// Base class
```

```
class Person {
public:
    void getPersonData() {
        cout << "Enter Name: ";
        cin >> name;
        cout << "Enter Age: ";
        cin >> age;
    }
    void displayPersonData() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }

protected:
    string name;
    int age;
};

// Derived class 1 (virtual inheritance)
class Student : virtual public Person {
public:
    void getStudentData() {
        cout << "Enter Student ID: ";
        cin >> studentID;
    }
    void displayStudentData() {
        cout << "Student ID: " << studentID << endl;
    }

protected:
    int studentID;
};
```

```
// Derived class 2 (virtual inheritance)
class Teacher : virtual public Person {
public:
    void getTeacherData() {
        cout << "Enter Subject: ";
        cin >> subject;
    }
    void displayTeacherData() {
        cout << "Subject: " << subject << endl;
    }

protected:
    string subject;
};

// Derived class that inherits from both Student and Teacher
class TeachingAssistant : public Student, public Teacher {
public:
    void getTeachingAssistantData() {
        getPersonData();   // Getting data from Person
        getStudentData();  // Getting data from Student
        getTeacherData();  // Getting data from Teacher
    }

    void displayTeachingAssistantData() {
        displayPersonData();   // Displaying data from Person
        displayStudentData();  // Displaying data from Student
        displayTeacherData();  // Displaying data from Teacher
    }
};
```

```
int main() {
    TeachingAssistant ta;

    cout << "Enter Teaching Assistant Data:" << endl;
    ta.getTeachingAssistantData();

    cout << "\nTeaching Assistant Details:" << endl;
    ta.displayTeachingAssistantData();

    return 0;
}
```

output:

```
Enter Teaching Assistant Data:
Enter Name: John
Enter Age: 25
Enter Student ID: 12345
Enter Faculty ID: 67890

Teaching Assistant Details:
Name: John
Age: 25
Student ID: 12345
Faculty ID: 67890
```

6.Write a program of constructors in Derived class.

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

```
// Base class
class Base {
public:
    // Base class constructor
    Base() {
        cout << "Base class constructor called" << endl;
    }

    // Base class parameterized constructor
    Base(int a) {
        cout << "Base class parameterized constructor called with value:
" << a << endl;
    }
};

// Derived class inheriting from Base class
class Derived : public Base {
public:
    // Derived class constructor
    Derived() {
        cout << "Derived class constructor called" << endl;
    }

    // Derived class parameterized constructor
    Derived(int x) : Base(x) {
        cout << "Derived class parameterized constructor called with
value: " << x << endl;
    }
};

int main() {
```

```
    cout << "Creating Derived object with default constructor:" << endl;
    Derived d1;  // Calls Base() and then Derived()

    cout << "\nCreating Derived object with parameterized constructor:"
<< endl;
    Derived d2(10);  // Calls Base(int) and then Derived(int)

    return 0;
}
```

output:

```
Creating Derived object with default constructor:
Base class constructor called
Derived class constructor called

Creating Derived object with parameterized constructor:
Base class parameterized constructor called with value: 10
Derived class parameterized constructor called with value: 10
```

7. Write a program that shows use of container.

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

int main() {
    // Create a vector of integers
```

```
    vector<int> numbers;

    // Adding elements to the vector
    numbers.push_back(10);
    numbers.push_back(20);
    numbers.push_back(30);
    numbers.push_back(40);

    // Display the elements of the vector
    cout << "Elements in the vector: ";
    for (int i = 0; i < numbers.size(); i++) {
        cout << numbers[i] << " ";
    }
    cout << endl;

    // Remove the last element from the vector
    numbers.pop_back();

    // Display the elements after removing one element
    cout << "After removing the last element: ";
    for (int i = 0; i < numbers.size(); i++) {
        cout << numbers[i] << " ";
    }
    cout << endl;

    // Using iterator to display elements
    cout << "Using iterator to display elements: ";
    for (vector<int>::iterator it = numbers.begin(); it != numbers.end();
++it) {
        cout << *it << " ";
    }
    cout << endl;
```

```
    return 0;
}
```

output:

```
Elements in the vector: 10 20 30 40
After removing the last element: 10 20 30
Using iterator to display elements: 10 20 30
```