Lab Number:	6
Student Name:	**Disha Maggu**
Roll No :	**09**

Title:

1. To perform Multiple Inheritance in C++. Create a student class representing student roll number, name and branch and an exam class (derived class of student) representing the scores of the student in various subjects (maths, physics and chemistry) and sports class representing the score in sports. The sports and exam class isinherited by a result class which adds the exam marks and sports score to generate the final result.

2. To perform Hierarchical Inheritance in C++. Create an Employee class with attributes EmpID and EmpSalary. Also create necessary methods/constructors to accept these values from the user. Create classes permenantEmployee and TemporaryEmployee which will be derived classes of Employee. Mention hike attribute in these derived classes and calculate the total salary using generate_salary() method for respective types of employees. Objects of the derived classes should be created and salaries for the permanent and temporary employees should be calculated and displayed on the screen.

Learning Objective:

- Students will be able to perform multiple inheritance using C++.

Learning Outcome:

- Understanding the inheritance concept and reusability of the code.

Course Outcome:

ECL304.2	Comprehend building blocks of OOPs language, inheritance, package and interfaces

Faculty: Ms. Deepali Kayande

Theory:

- Explain in details about inheritance, its types, syntaxes and block diagrams.
  Inheritance is the process of creating a new Class, called the **Derived Class** , from the existing class, called the **Base Class** . The Inheritance has many advantages, the most important of them being the reusability of code. Rather than developing **new Objects** from scratch, new code can be based on the work of other developers, adding only the new features that are needed. The reuse of **existing classes** saves time and effort.

Types of inheritance:

✦ Single Inheritance - When a **Derived Class** to inherit properties and behavior from a single **Base Class** , it is called as single inheritance.

Syntax - *class Base*
{
// BODY OF THE BASE CLASS
};
class Derived : acess_specifier Base
{
// BODY OF THE DERIVED CLASS
};

✦ Multi Level Inheritance - A **derived class** is created from another derived class is called **Multi Level Inheritance** .

Syntax :

class A // Base class

{

// BODY OF CLASS A

};

class B : acess_specifier A // Derived class of A

{

// BODY OF CLASS B

};

class C : access_specifier B // Derived from derived class B

{

// BODY OF CLASS C

};

- Hierarchical Inheritance - More than one **derived classes** are created from a single base class, is called **Hierarchical Inheritance** .
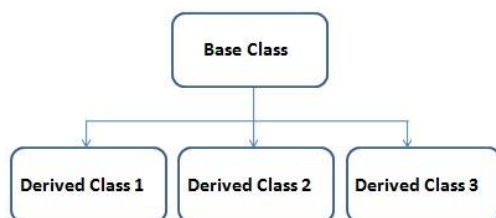
  Syntax :

  class A // Base class of B

  {

  // BODY OF THE PROGRAM

  };

  class B : access_specifier A // Derived class of A

  {

  // BODY OF THE PROGRAM

  };

  class C : access_specifier A // Derived class of A

  {

  // BODY OF THE PROGRAM

  };

  class D : access_specifier A // Derived class of A

  {

  // BODY OF THE PROGRAM

  };

- Hybrid Inheritance - Any combination of above **three inheritance** (single, hierarchical and multi level) is called as **hybrid inheritance** .

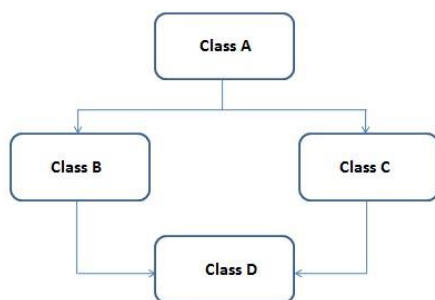**Syntax :**

class A

{

// BODY OF THE CLASS A

};

class B : public A

{

// BODY OF THE CLASS A

};

class C

{

// BODY OF THE CLASS A

};

class D : public B, public C

{

// BODY OF THE CLASS A

};

- Multiple Inheritance - Multiple inheritances allows programmers to create classes that combine aspects of multiple classes and their corresponding hierarchies. **Syntax -**

  class A // Base class of B

  {

```
 // BODY OF THE CLASS A

  };

  class B // Derived class of A and Base class

  {

 // BODY OF THE CLASS B

  };

 class C : acess_specifier A, access_specifier A // Derived class of A and B

 {

  // BODY OF CLASS C

  };
```

Algorithm 1 :

1- Create a parent class student and initialize its data members.

2- Create the derived class of student class - exam class to take input of marks

3- Create sports class to take input of marks

4- Create the result class to inherit the exam class and sports class publically and to calculate the total.

5- Create the main function to call the class functionality and display the results

Program 1:

```cpp
#include<iostream>

using namespace std;

class student{

        public:

                int roll_number;

                string name;

                string branch;

                student(){

                cout<<"Enter roll number"<<endl;

                cin>>roll_number;

                cout<<"Enter name"<<endl;

                cin>>name;

                cout<<"Enter branch"<<endl;

                cin>>branch;

}

};

 class exam: public student{

        public:

        int maths;

        int physics;

        int chemistry;

        exam(){
```

```cpp
        cout<<"Enter Physics Marks"<<endl;

        cin>>physics;

        cout<<"Enter chemistry Marks"<<endl;

        cin>>chemistry;

        cout<<"Enter Maths Marks"<<endl;

        cin>>maths;

    }

};


class sport{

        public:

        int sports;

        sport(){

        cout<<"Enter sports Marks"<<endl;

        cin>>sports;

    }

};


class result : public exam, public sport {

        public:

                int total;

                result(){

                total = maths + physics + chemistry + sports;
```

```cpp
            cout<<total<<endl;

        }

};



int main()

{

    result obj;

    return 0;

}
```
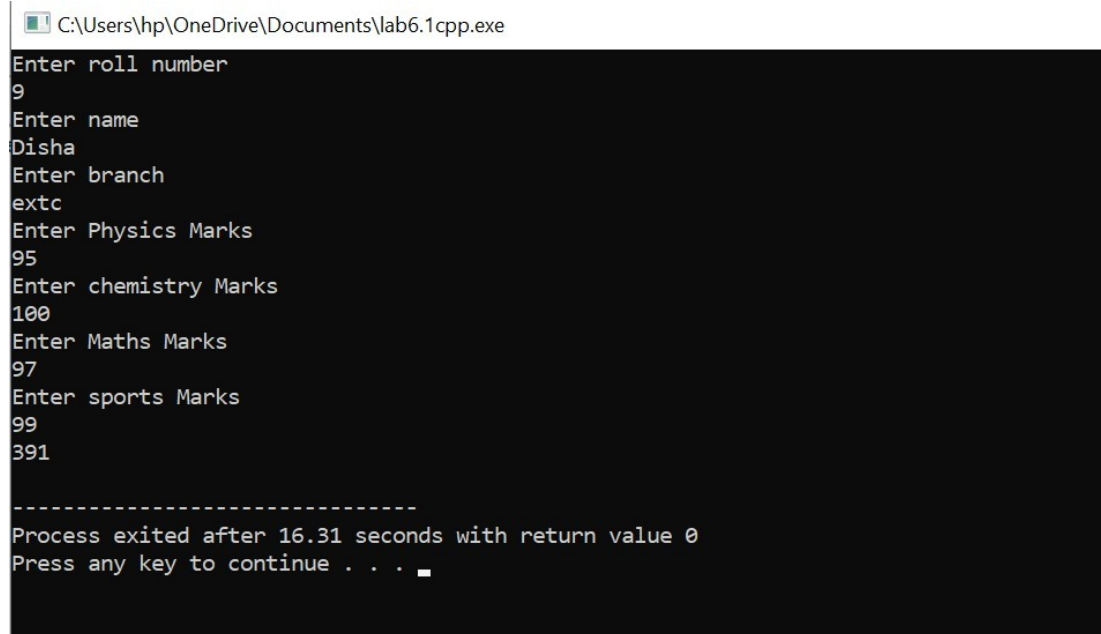
Input Given:

ROLL NO	9
NAME	DISHA
BRANCH	EXTC
PHYSICS MARKS	95
CHEMISTRY MARKS	100
MATHS MARKS	97
SPORTS MARKS	99

Faculty: Ms. Deepali Kayande

Output:


```
C:\Users\hp\OneDrive\Documents\lab6.1cpp.exe
Enter roll number
9
Enter name
Disha
Enter branch
extc
Enter Physics Marks
95
Enter chemistry Marks
100
Enter Maths Marks
97
Enter sports Marks
99
391

------------------------------
Process exited after 16.31 seconds with return value 0
Press any key to continue . . . _
```

Algorithm 2:

1. Creating the parent class employee and initialize its data members.(EmpId ,Emp Salary) and a basic function get details() to print the details.

2. Create 2 child class permanent employee and temporary employee that inherit employee class publically.

3. In this classes , create generate salary() that return the employee salary + hike in their salary

4. In main function, Create the object of derived class and print their respective details.

Program 2:

```cpp
#include <iostream>

using namespace std;


class Employee

{

        protected:

        string EmpID = "";



        double Empsalary;



        public:

                Employee()

                {

        cout<<endl<<"Enter the employee id : ";

        cin>>EmpID;

                        cout<<endl<<"Enter the employee Sal : ";

        cin>>Empsalary;



                }



        void getDetails()

        {

                cout <<endl<< "EmployeeID is : " << EmpID;
```

```cpp
                cout <<endl<<endl<<"Employee Total Salary is : " << Empsalary;

        }



};



class Permanent_Employee: public Employee

{

        double hike;


        public:


        Permanent_Employee( double increment)

        {

                        hike = increment;

        }


    void getDetails()

        {

                cout <<endl<< "EmployeeID is : " << EmpID;

                cout <<endl<<endl<<"Employee Total Salary is : " <<
generate_salary();

        }
```

```cpp
        float generate_salary()

        {

                return (Empsalary + hike);

        }

};


class Temporary_Employee: public Employee

{

        double hike;


        public:


        Temporary_Employee( double increment)

        {

                        hike = increment;

        }


    void getDetails()

        {

                cout <<endl<< "EmployeeID is : " << EmpID;

                cout <<endl<<endl<<"Employee Total Salary is : " <<
generate_salary();

        }
```

```cpp
        float generate_salary()

        {

                return (Empsalary + hike);

        }

};


int main()

{


    cout<<endl<<"For Temporary Employee: "<<endl;

        Temporary_Employee T(667.6);

    cout << endl << endl << "Details of Temporary_Employee : " << endl;

        T.getDetails();

    cout<<endl<<endl<<"For Permanent Employee: "<<endl;

        Permanent_Employee P(777.99);

        cout << endl << endl << "Details of Permanent_Employee : " << endl;

        P.getDetails();


        return 0;

}
```

Input Given:

For temporary employee:	For permanent employee:
Employee id: 102	Employee id: 222
Employee sal: 8000	Employee sal:15000

Output:


```
C:\Users\hp\OneDrive\Documents\lab6.2.exe

For Temporary Employee:

Enter the employee id : 102

Enter the employee Sal : 8000


Details of Temporary_Employee :

EmployeeID is : 102

Employee Total Salary is : 8667.6

For Permanent Employee:

Enter the employee id : 222

Enter the employee Sal : 15000


Details of Permanent_Employee :

EmployeeID is : 222

Employee Total Salary is : 15778
-------------------------------
Process exited after 31.88 seconds with return value 0
Press any key to continue . . .
```

Faculty: Ms. Deepali Kayande