| **Lab Number:** | **6** |
| --- | --- |
| **Student Name:** | **Omkar Santoah Mundhe** |
| **Roll No :** | **20** |

# 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++.

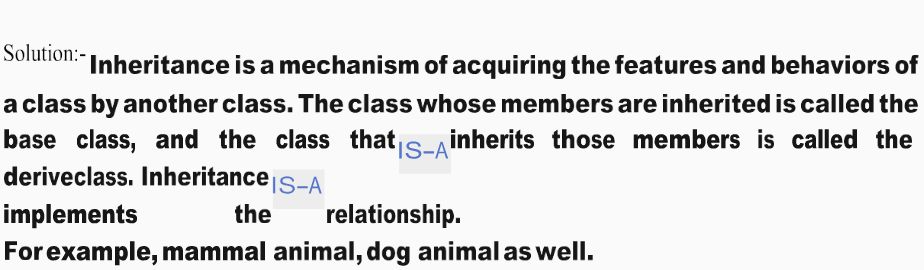
# Learning Outcome:

* + Understanding the inheritance concept and reusability of the code.

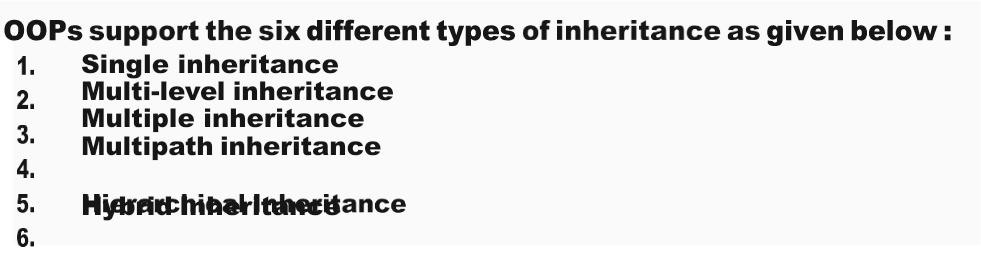
# Course Outcome:

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

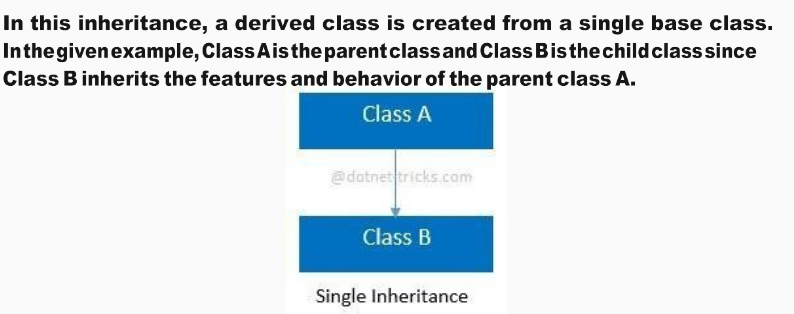
**Theory:**

* + Explain in details about inheritance, its types, syntaxes and block diagrams.

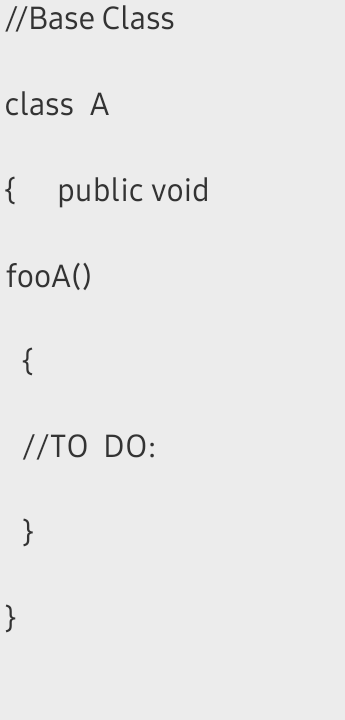
| IS-A | mammal; Hencedog | IS-A |
| --- | --- | --- |

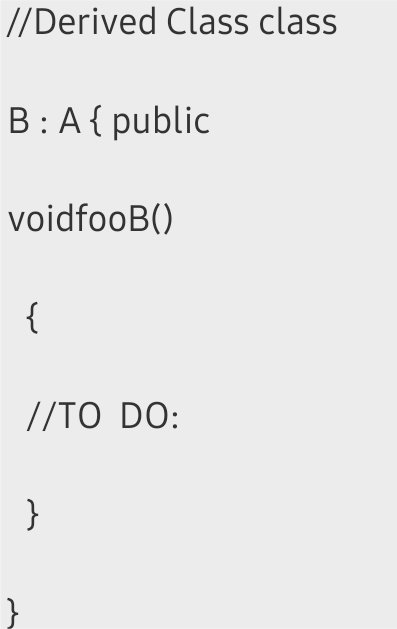


1. Single inheritance

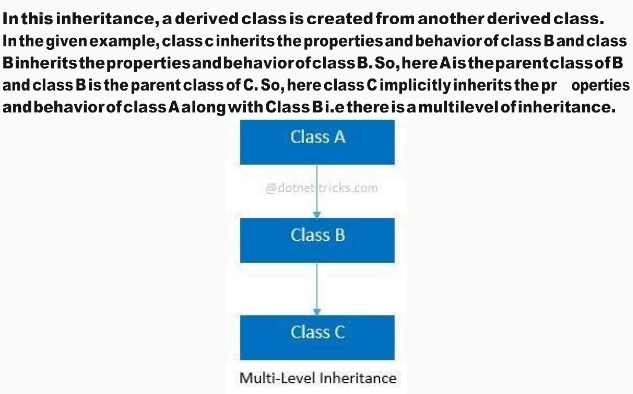


Syntax for Single Inheritance

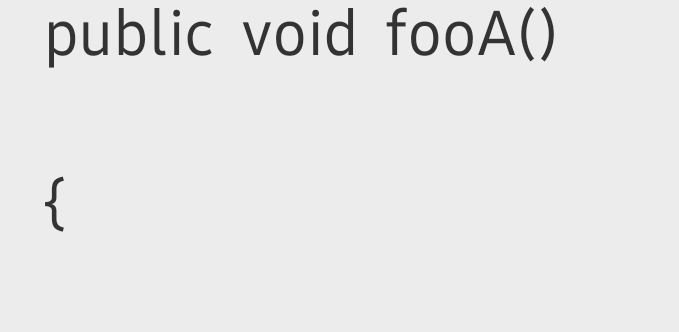
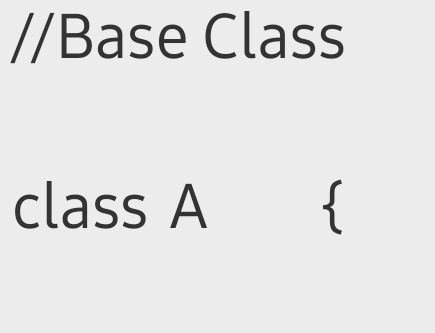


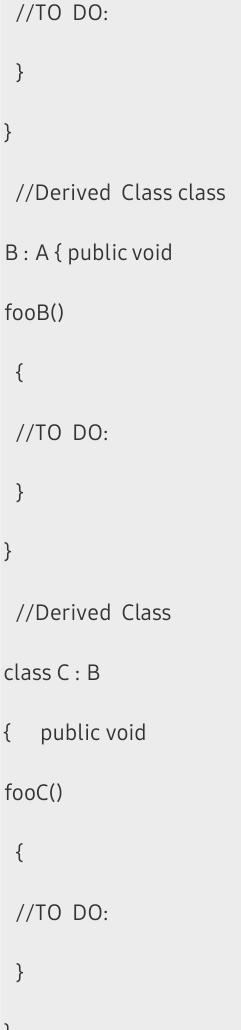


1. Multi-level inheritance

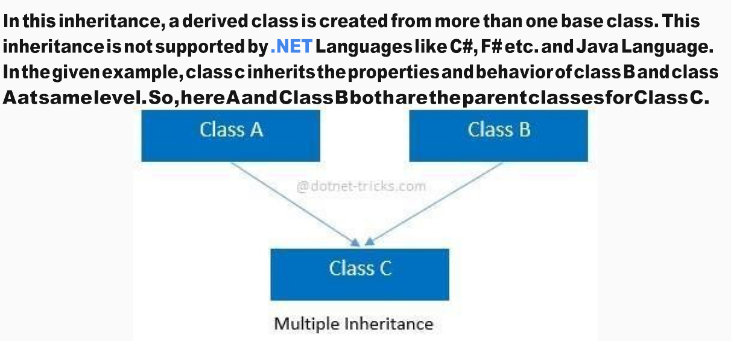


Syntax for Multi-level Inheritance

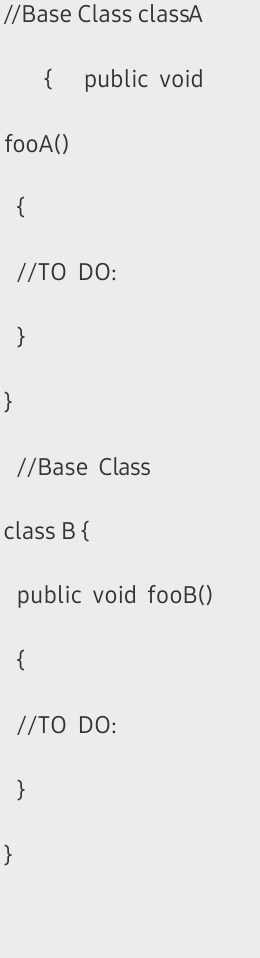


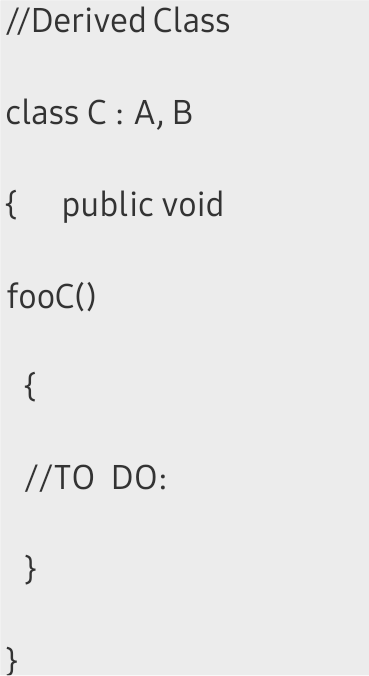


1. Multiple inheritance

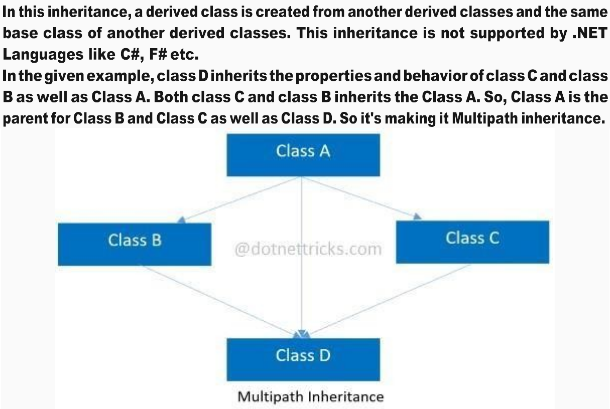


Syntax for Multiple Inheritance



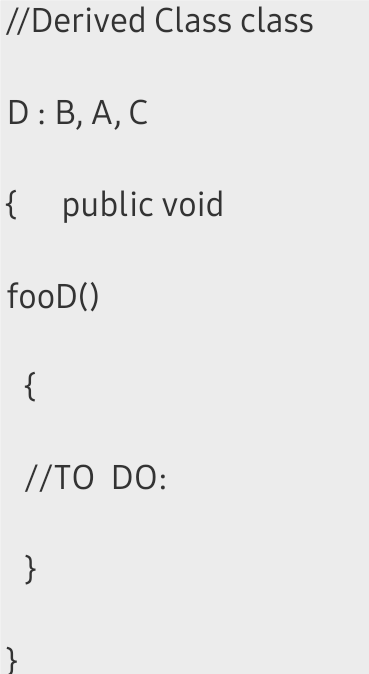


1. Multipath inheritance

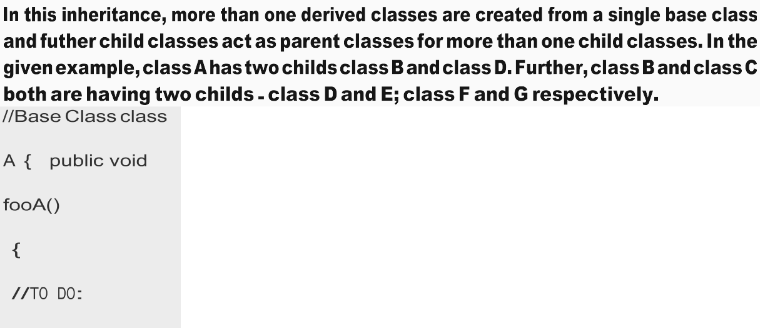


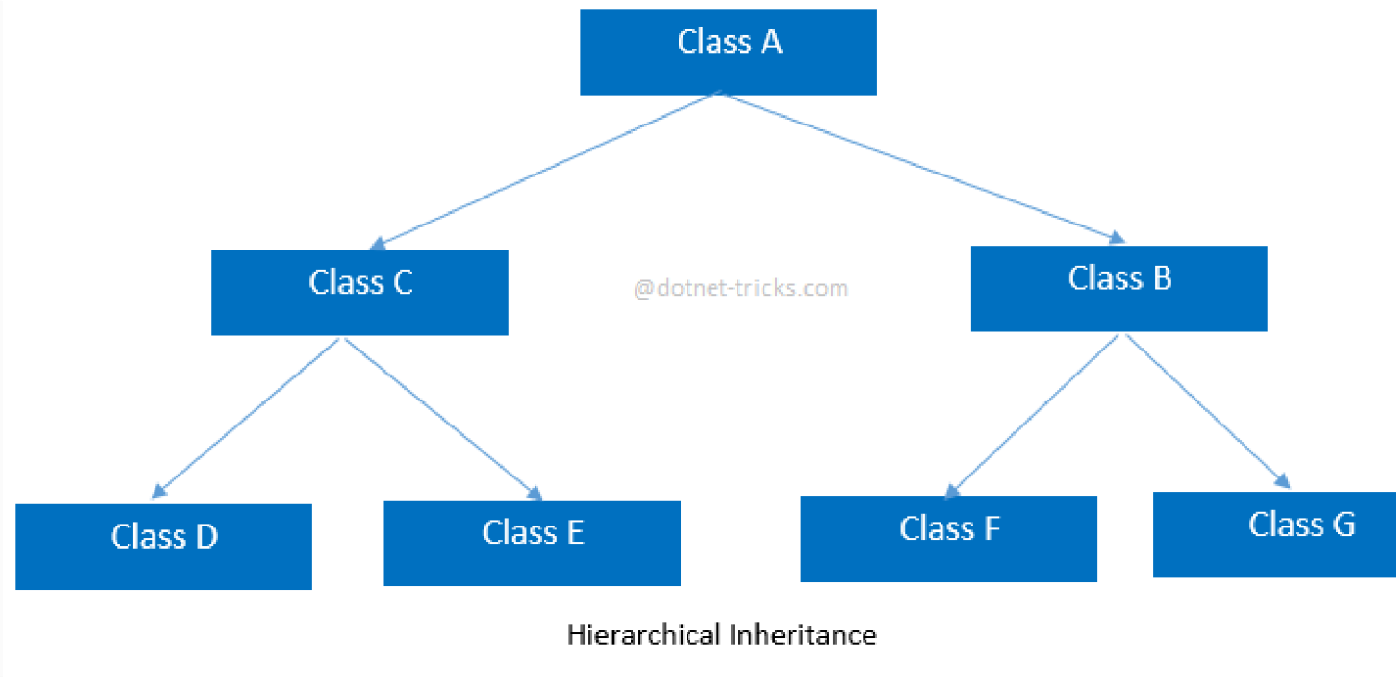
Syntax for Multipath Inheritance



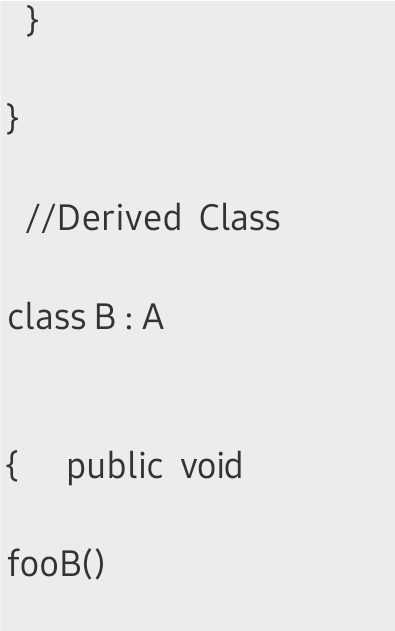


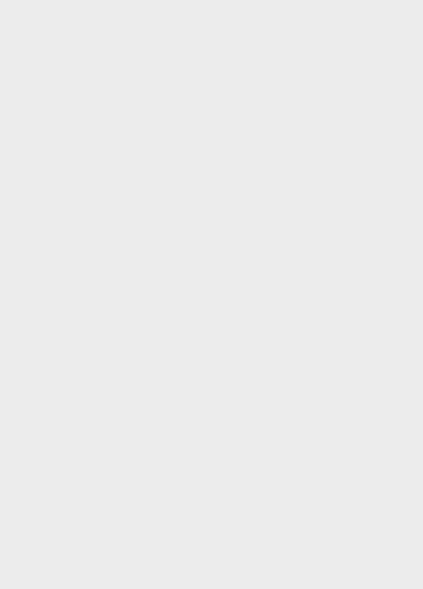
1. Hierarchical Inheritance





Syntax for Hierarchical Inheritance





{

//TO DO:

}

}

//Derived Class class C : A

{ public void fooC()

{

//TO DO:

}

}

//Derived Class class D : C

{ public void fooD()

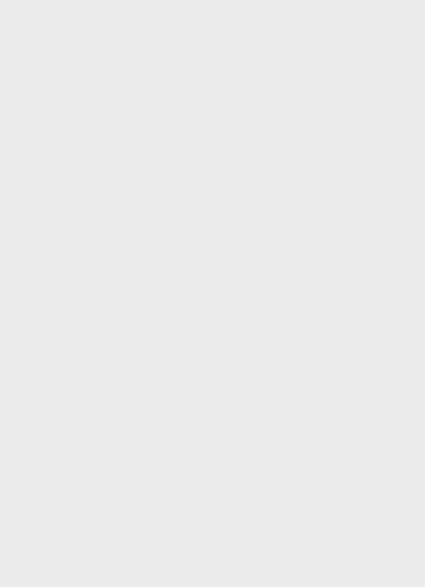
{

//TO DO:

}

}

//Derived Class



class E : C

{ public void fooE()

{

//TO DO:

}

}

//Derived Class class F : B

{ public void fooF()

{

//TO DO:

}

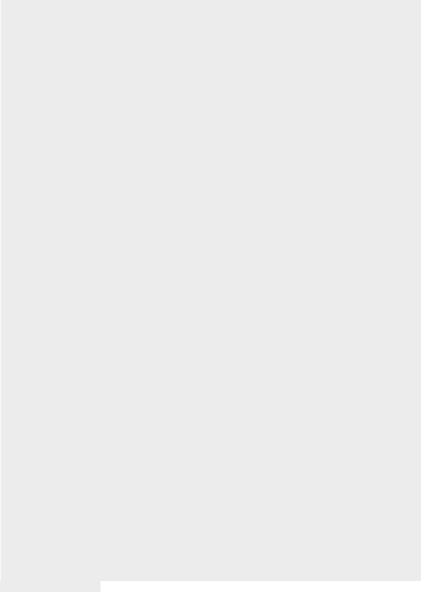
}

//Derived Class class G :B

{ public void fooG()

{

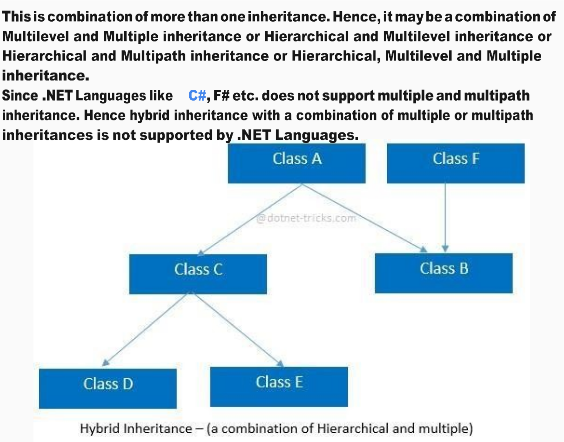
//TO DO:



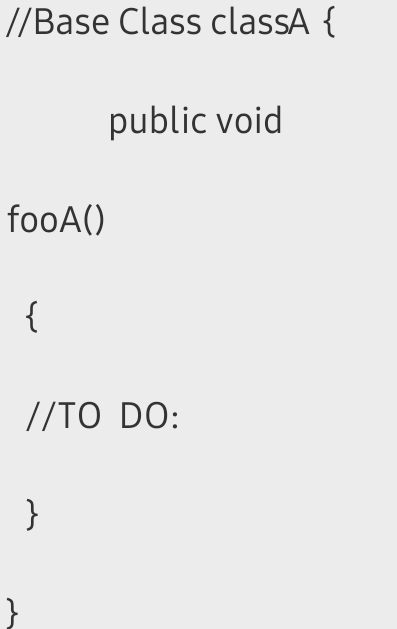
}

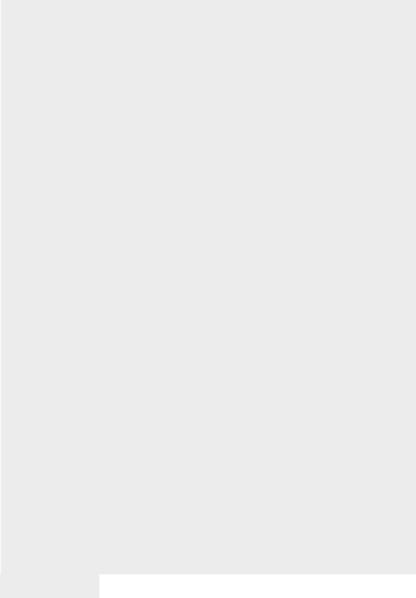
}

1. Hybrid inheritance



Syntax for Hybrid Inheritance





//Base Class class F { public void fooF()

{

//TO DO:

}

}

//Derived Class class B : A, F

{ public void fooB()

{

//TO DO:

}

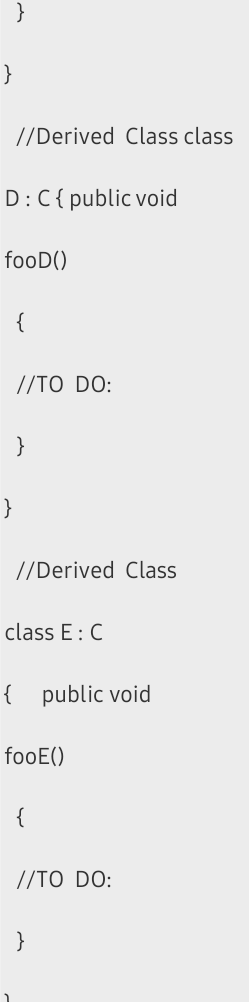
}

//Derived Class class C : A

{ public void fooC()

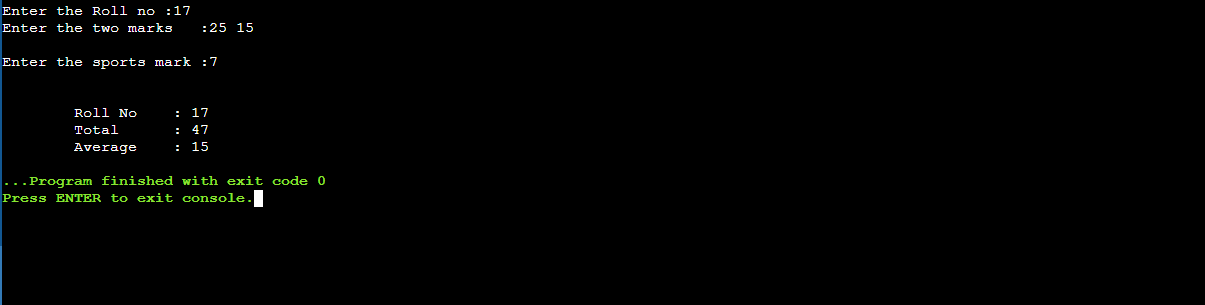
{

//TO DO:



# 6.0(Multiple) :- Input given a=17 b= 25 15

# OUTPUT:



**6.1(**Hierarchical) **: - Input given:- a= ygh b= 785 c=M**

**OUTPUT:**

