AIR UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EXPERIMENT NO 09

Lab Title: Multiple Inheritance

Student Name: _____ **Reg. No:** _____

Objective: _____

_____

LAB ASSESSMENT:

Attributes	Excellent (5)	Good (4)	Average (3)	Satisfactory (2)	Unsatisfactory (1)
Ability to Conduct Experiment					
Ability to assimilate the results					
Effective use of lab equipment and follows the lab safety rules					

Total Marks: _____ Obtained Marks: _____

LAB REPORT ASSESSMENT:

Attributes	Excellent (5)	Good (4)	Average (3)	Satisfactory (2)	Unsatisfactory (1)
Data presentation					
Experimental results					
Conclusion					

Total Marks: _____ Obtained Marks: _____

Date: _____ Signature: _____

EXPERIMENT NO 9

Multiple Inheritance

Objectives

- To learn what is Multiple Inheritance
- Understand and solve the diamond problem due to Multiple Inheritance

Equipment required:

- Visual Studio/ Dev C++

Description

Inheritance is the process of inheriting properties of objects of one class by objects of another class. The class which inherits the properties of another class is called **Derived or Child or Sub class** and the class whose properties are inherited is called **Base or Parent or Super class**. When a class is derived from two or more base classes, such inheritance is called **Multiple Inheritance**. It allows us to combine the features of several existing classes into a single class.

For example,

- Petrol is derived from both liquid and fuel.
- A child has character of both his/her father and mother, etc

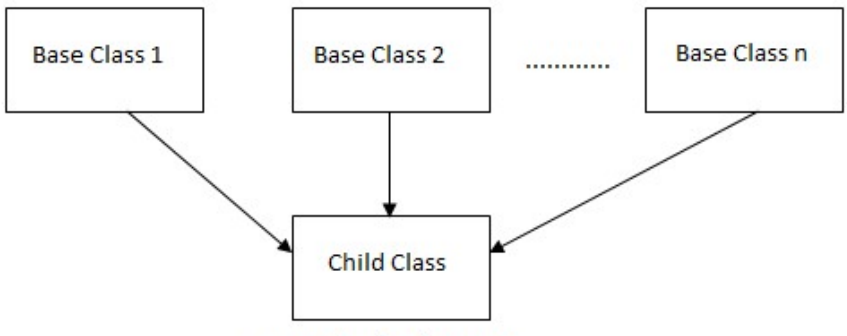

Fig: Multiple Inheritance

Syntax of Multiple Inheritance

```
class A
{
  ..........
};
class B
{
  ..........
 } ;
class C : access_specifier A, access_specifier B              // derived class from A and B
{
  ..........
} ;
```

Example

```
#include <iostream>
using namespace std;
class A{
          public:
          int x;
          void getx(){
             cout << "enter value of x: ";       cin >> x;   }
};
class B{
          public:
          int y;
          void gety(){
             cout << "enter value of y: ";        cin >> y;   }
};
class C : public A, public B{          //C is derived from class A and class B
          public:
          void sum(){
             cout << "Sum = " << x + y;       }
};

int main(){
           C obj1;                          //object of derived class C
           obj1.getx();
           obj1.gety();
           obj1.sum();
           return 0;    }
```

Output

```
enter value of x: 5
enter value of y: 7
Sum = 12
```

Ambiguity in Multiple Inheritance

The most obvious problem with multiple inheritance occurs during function overriding.

Suppose, two base classes have a same function which is not overridden in derived class.

If you try to call the function using the object of the derived class, compiler shows error. It's because compiler doesn't know which function to call. For example,

```
class base1 {
  public:
     void someFunction( ) {....}
};
class base2 {
   void someFunction( ) {....}
};
class derived : public base1, public base2 {};

int main() {
   derived obj;
   obj.someFunction() // Error!
}
```

This problem can be solved using the scope resolution function to specify which function to class either base1or base2.

```
int main() {
   obj.base1::someFunction( );  // Function of base1 class is called
   obj.base2::someFunction();   // Function of base2 class is called.
}
```

The diamond problem

The diamond problem occurs when two super classes of a class have a common base class. For example, in the following diagram, the TA class gets two copies of all attributes of Person class, this causes ambiguities.

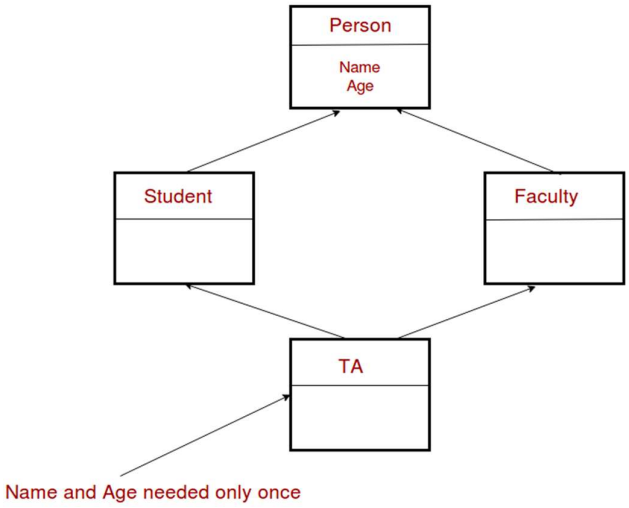

Figure 1

LAB TASK

1. Write a class **Teacher** that contains the following data members.
    - Institution
    - Salary
    - Experience

The class **Teacher** contain the member functions to input and display the data members.

Write another class **Writer** that contains the following data members.

- Number of books written

The class **Writer** contains the member functions to input and display the data members.

Write a third-class **Scholar** that inherits both Teacher and Writer classes. Define a relationship between classes using **public** inheritance.

The class **Scholar** contains the following data members.

- Name
- Age
- Address

The class **Scholar** contains the member functions to input and display the data members.

```
class teacher{
        char institution[10];
        int salary, experience;
        int age;
        public:
                void input(){ }
                void display(){ }          };

class writer{
        int book;
        public:
                void input(){ }
                void display(){ }          };

class scholar: public teacher, public writer{
        char name[50], address[50];
        int age;
        public:
                void input(){ }
                void display(){ }                  };

int main(){

        return 0;       }
```

TEST PLAN

Commands for main function
Scholar S;
S.input();
S.teacher::input();
S.writer::input();
S.display();
S.teacher::display();
S.writer::display();

2. Write a class **Student** that contains the following data members.
    - Name
    - Roll no.

   The class **Student** contains the following member function.

    - A **default constructor** to initialize the value of data members.
    - An **input function** to take the value of data members from user.

   Write a class **marks** that contains the following data members.
    - Array that stores marks of 6 subjects
    - A variable s to store sum of marks

   The class **marks** contain the following member function.

    - A **default constructor** to initialize the value of data members.
    - An **input function** to take the marks from user.
    - A **sum function** to calculate the sum of marks of all six subjects.
        - ✓ This function also returns the value of sum.

Write a class **result** that inherits the student and marks classes. Define a relationship between classes using **public** inheritance.

The class **result** contains the following data members.

- Avg (A variable to store average of marks)

The class **result** contains the following member function.

- A **default constructor** to initialize the value of data members.
- An **input function** to get input.
    - ✓ Calling Student::input(); inside this input function
    - ✓ Calling marks::input(); inside this input function
- An **avg_marks function** to calculate the average of marks of all six subjects.
    - ✓ Calling marks::sum() inside this function and divide it by 6

- ✓ Hint : (cout << marks::sum()/6;)

```
class student{
        // data members
        public:
                // default constructor
                // input function
};

class mark{
        // data members
        public:
                // default constructor
                // input function
                // sum function
};

class result:  access_specifier  student, access_specifier mark{
        // data member

        public:
                // default constructor
                // input function
                // avg_marks function
};

int main(){
        return 0;
}
```

TEST PLAN

Commands for main function
result r;
r.input();
r.avg_marks();

3. Implement the **Example of TA class** shown above in figure 1 using multiple inheritance
   - With diamond problem
   - Without diamond problem
   - Notice the difference among two.

   Help : https://www.geeksforgeeks.org/multiple-inheritance-in-c/