Name: Om Kshirsagar　　　　　　　　　　　　　　　　　　　　　　　　Subject: OOPS
Class: FYIT　　　　　　　　　　　　　　　　　　　　　　　　　　　　Date: 2/3/2021

Practical 3: Inheritance

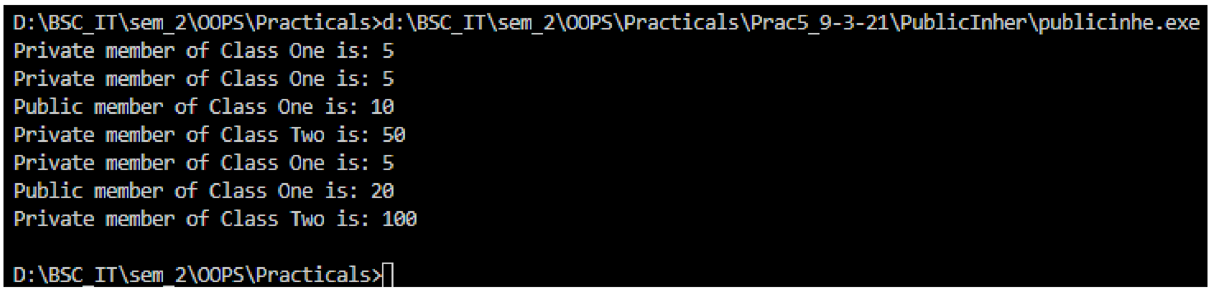
A. Design a class for single level inheritance using public derivation.

Code:

```
// Design a class for single level inheritance using public derivation.
#include<iostream>

using std::cout;
using std::endl;
//-----------Class Definitions-------------------
class One
{
private:
    int membOnePr;
public:
    int membOnePb;
    One() { }
    void initializeBothMembOfOne();
    int getMembOnePr();
    void showMembOnePr();
    ~One(){ }
};

class Two: public One
{
private:
    int membTwoPr;
public:
    Two(){ }
    void multi();
    void disp();
    ~Two(){ }
};
//----------------Class One functions------------------
void One::initializeBothMembOfOne()
{
    membOnePr = 5, membOnePb = 10;
}
int One::getMembOnePr()
{
    return membOnePr;
}
void One::showMembOnePr()
{
    cout << "Private member of Class One is: "<<membOnePr<<endl;
}
//------------------Class Two functions---------------------
```

```
void Two::multi()
{
    membTwoPr = membOnePb * getMembOnePr();
}
void Two::disp()
{
    showMembOnePr();
    cout << "Public member of Class One is: "<<membOnePb<<endl;
    cout << "Private member of Class Two is: "<<membTwoPr<<endl;
}
//------------------Main Declaration--------------------------
int main()
{
    Two two;
    two.initializeBothMembOfOne();
    two.multi();
    two.showMembOnePr();
    two.disp();

    two.membOnePb = 20;
    two.multi();
    two.disp();

    return 0;
}
```

Output:



```
D:\BSC_IT\sem_2\OOPS\Practicals>d:\BSC_IT\sem_2\OOPS\Practicals\Prac5_9-3-21\PublicInher\publicinhe.exe
Private member of Class One is: 5
Private member of Class One is: 5
Public member of Class One is: 10
Private member of Class Two is: 50
Private member of Class One is: 5
Public member of Class One is: 20
Private member of Class Two is: 100

D:\BSC_IT\sem_2\OOPS\Practicals>
```

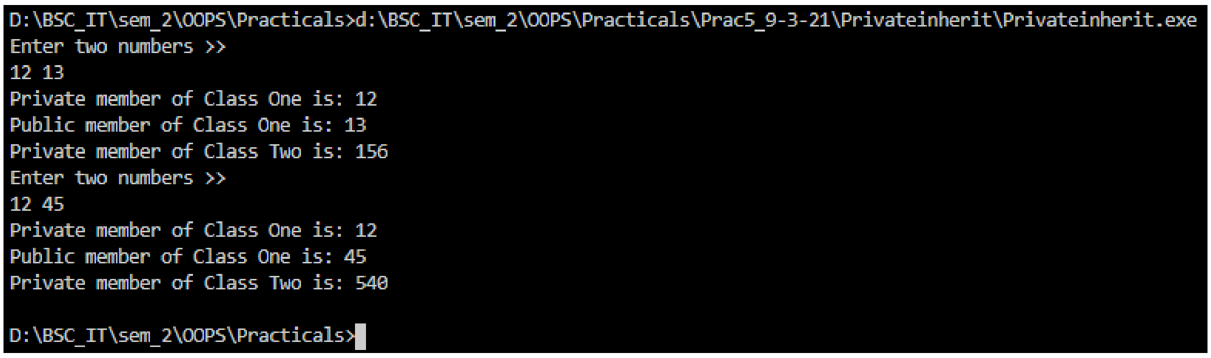
B. Design a class for single level inheritance using private derivation.

Code:

```
// Design a class for single level inheritance using private derivation.
#include<iostream>

using std::cout;
using std::endl;
//-----------Class Definations-------------------
class One
{
private:
    int membOnePr;
```

```cpp
public:
    int membOnePb;
    One() { }
    void initializeBothMembOfOne();
    int getMembOnePr();
    void showMembOnePr();
    ~One(){ }
};

class Two: private One
{
private:
    int membTwoPr;
public:
    Two(){ }
    void multi();
    void disp();
    ~Two(){ }
};
//----------------Class One functions------------------
void One::initializeBothMembOfOne()
{
    cout << "Enter two numbers >> \n";
    std::cin >> membOnePr >> membOnePb;
}
int One::getMembOnePr()
{
    return membOnePr;
}
void One::showMembOnePr()
{
    cout << "Private member of Class One is: "<<membOnePr<<endl;
}
//------------------Class Two functions---------------------
void Two::multi()
{
    initializeBothMembOfOne();
    membTwoPr = membOnePb * getMembOnePr();
}
void Two::disp()
{
    showMembOnePr();
    cout << "Public member of Class One is: "<<membOnePb<<endl;
    cout << "Private member of Class Two is: "<<membTwoPr<<endl;
}
//------------------Main Declaration--------------------------
int main()
{
    Two two;
    two.multi();
    two.disp();

    two.multi();
    two.disp();

    return 0;
}
```

Output:



```
D:\BSC_IT\sem_2\OOPS\Practicals>d:\BSC_IT\sem_2\OOPS\Practicals\Prac5_9-3-21\Privateinherit\Privateinherit.exe
Enter two numbers >>
12 13
Private member of Class One is: 12
Public member of Class One is: 13
Private member of Class Two is: 156
Enter two numbers >>
12 45
Private member of Class One is: 12
Public member of Class One is: 45
Private member of Class Two is: 540

D:\BSC_IT\sem_2\OOPS\Practicals>
```

C. Design a class for multilevel inheritance.

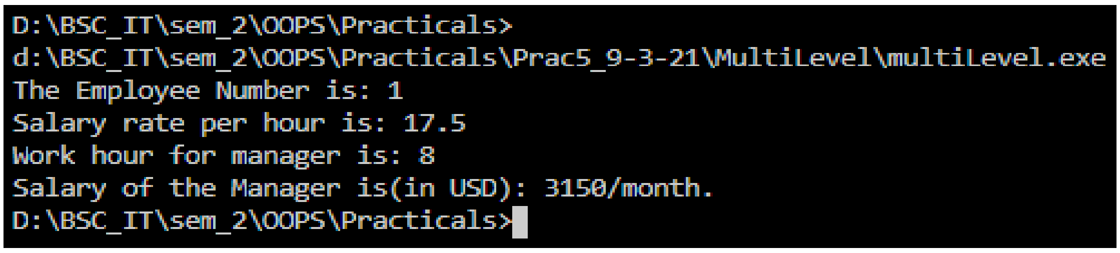
Code:

```
// Design a class for multilevel inheritance.
#include<iostream>

using std::cout;
using std::endl;
//------------------Employee Class------------------
class employee
{
protected:
    int employeeNo;
public:
    employee(){ }
    void initializeENO(int x){
        employeeNo = x;
    }
    void printENO(){
        cout << "The Employee Number is: " << employeeNo <<endl;
    }
    ~employee(){ }
};
//-----------Manager Class(Level 1)-----------------
class manager: public employee
{
protected:
```

Name: Om Kshirsagar Subject: OOPS
Class: FYIT Date: 2/3/2021

```
    float salaryRate;
    int workHours;
public:
    manager(){ }
    void initializeMGER(float x, int y){
        salaryRate = x;
        workHours = y;
    }
    void printMGER(){
        cout << "Salary rate per hour is: " << salaryRate <<endl;
        cout << "Work hour for manager is: " << workHours <<endl;
    }
    ~manager(){ }
};
//-------------Total Class(Level 3)---------------------
class TotalSal: public manager
{
protected:
    float total;
public:
    TotalSal(){ }
    void printTTL(){
        total = salaryRate * workHours *22.5;
        printENO();
        printMGER();
        cout << "Salary of the Manager is(in USD): " << total <<"/month."
;
    }
    ~TotalSal(){ }
};
//---------------Main Program------------------------------
int main()
{
    TotalSal Manager1;
    Manager1.initializeENO(1);
    Manager1.initializeMGER(17.5, 8);

    Manager1.printTTL();
    return 0;
}
```

Output:



```
D:\BSC_IT\sem_2\OOPS\Practicals>
d:\BSC_IT\sem_2\OOPS\Practicals\Prac5_9-3-21\MultiLevel\multiLevel.exe
The Employee Number is: 1
Salary rate per hour is: 17.5
Work hour for manager is: 8
Salary of the Manager is(in USD): 3150/month.
D:\BSC_IT\sem_2\OOPS\Practicals>
```

Name: Om Kshirsagar Subject: OOPS
Class: FYIT Date: 2/3/2021

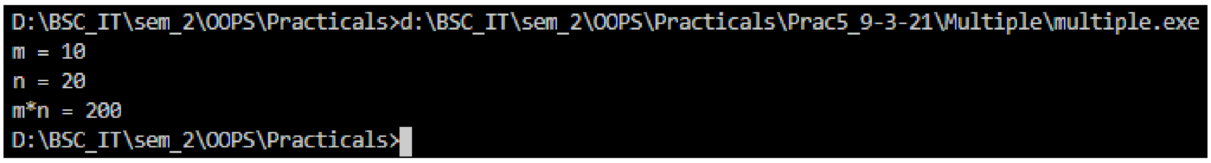
D. Design a class to implement Multiple Inheritance.

Code:

```cpp
// Design a class to implement Multiple Inheritance.
#include<iostream>
using std::endl;
using std::cout;
//---------------M Class-------------------
class M
{
protected:
    int m;
public:
    void get_m(int);
};
//---------------N Class-------------------
class N
{
protected:
    int n;
public:
    void get_n(int);
};
//--------------P Class---------------------
class P : public M, public N
{
public:
    void display(void);
};
void M :: get_m(int x)
{
    m=x;
}
void N :: get_n(int y)
{
    n=y;
}
void P :: display(void)
{
    cout << "m = " << m << endl;
    cout << "n = " << n << endl;
    cout << "m*n = " << m*n;
}
//---------Main Program-----------------
int main()
{
   P p;
```

```
    p.get_m(10);
    p.get_n(20);
    p.display();

    return 0;
}
```

Output:



```
D:\BSC_IT\sem_2\OOPS\Practicals>d:\BSC_IT\sem_2\OOPS\Practicals\Prac5_9-3-21\Multiple\multiple.exe
m = 10
n = 20
m*n = 200
D:\BSC_IT\sem_2\OOPS\Practicals>
```

E. Design a class to implement Hybrid Inheritance.

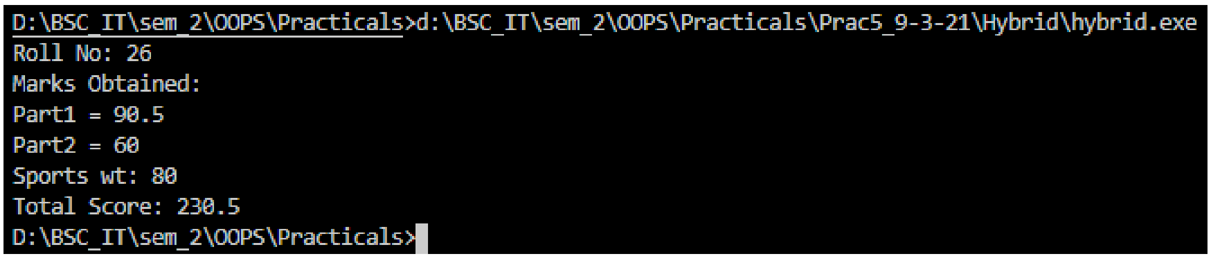
Code:

```
// class to implement Hybrid Inheritance.
#include<iostream>
using std::endl;
using std::cout;
//----------------------Student Class-------------------------
class student
{
protected:
    int roll_number;
public:
   void get_number(int a)
   {
       roll_number = a;
   }
   void put_number(void)
   {
       cout << "Roll No: " << roll_number << endl;

   }
};
//-----------------------Test Class----------------------------
class test : public student
{
protected:
    float part1, part2;
public:
    void get_marks(float x, float y)
    {
        part1 = x; part2 = y;
    }
    void put_marks(void)
    {
```

```cpp
        cout << "Marks Obtained: " << endl;
        cout << "Part1 = " << part1 << endl;
        cout << "Part2 = " << part2 << endl;
    }
};
//---------------------------Sports Class----------------------
class sports
{
protected:
    float score;
public:
    void get_score(float s)
    {
        score = s;
    }
    void put_score(void)
    {
        cout << "Sports wt: " << score << endl;
    }
};
//-------------------Result Clasx----------------------
class result : public test, public sports
{
    float total;
public:
    void display(void);
};

void result :: display(void)
{
    total = part1 + part2 + score;

    put_number();
    put_marks();
    put_score();

    cout << "Total Score: " << total;
}
//----------------------Main Program------------------------------
int main()
{
    result student_1;
    student_1.get_number(26);
    student_1.get_marks(90.5, 60.0);
    student_1.get_score(80.0);
    student_1.display();
    return 0;
}
```

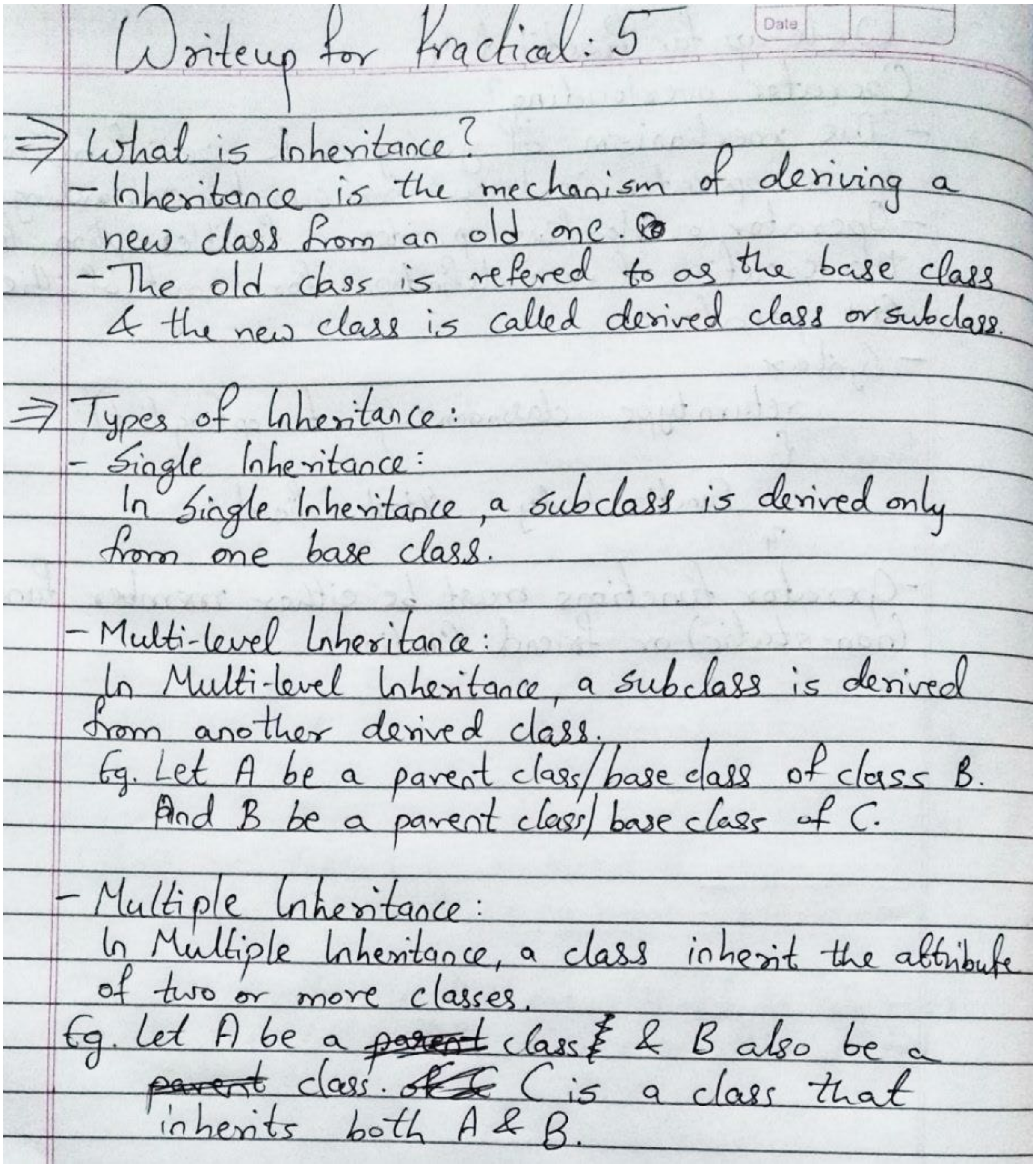
Name: Om Kshirsagar Subject: OOPS
Class: FYIT Date: 2/3/2021

Output:



```
D:\BSC_IT\sem_2\OOPS\Practicals>d:\BSC_IT\sem_2\OOPS\Practicals\Prac5_9-3-21\Hybrid\hybrid.exe
Roll No: 26
Marks Obtained:
Part1 = 90.5
Part2 = 60
Sports wt: 80
Total Score: 230.5
D:\BSC_IT\sem_2\OOPS\Practicals>
```

Write-up:



Writeup for Practical: 5

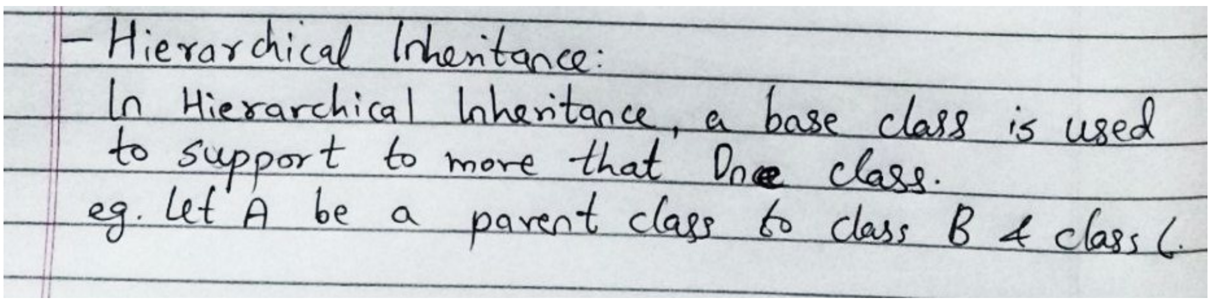
⇒ What is Inheritance?
- Inheritance is the mechanism of deriving a new class from an old one.
- The old class is refered to as the base class & the new class is called derived class or subclass.

⇒ Types of Inheritance:
- Single Inheritance:
In Single Inheritance, a subclass is derived only from one base class.

- Multi-level Inheritance:
In Multi-level Inheritance, a subclass is derived from another derived class.
Eg. Let A be a parent class/base class of class B.
And B be a parent class/base class of C.

- Multiple Inheritance:
In Multiple Inheritance, a class inherit the attribute of two or more classes.
Eg. Let A be a ~~parent~~ class & B also be a ~~parent~~ class. C is a class that inherits both A & B.

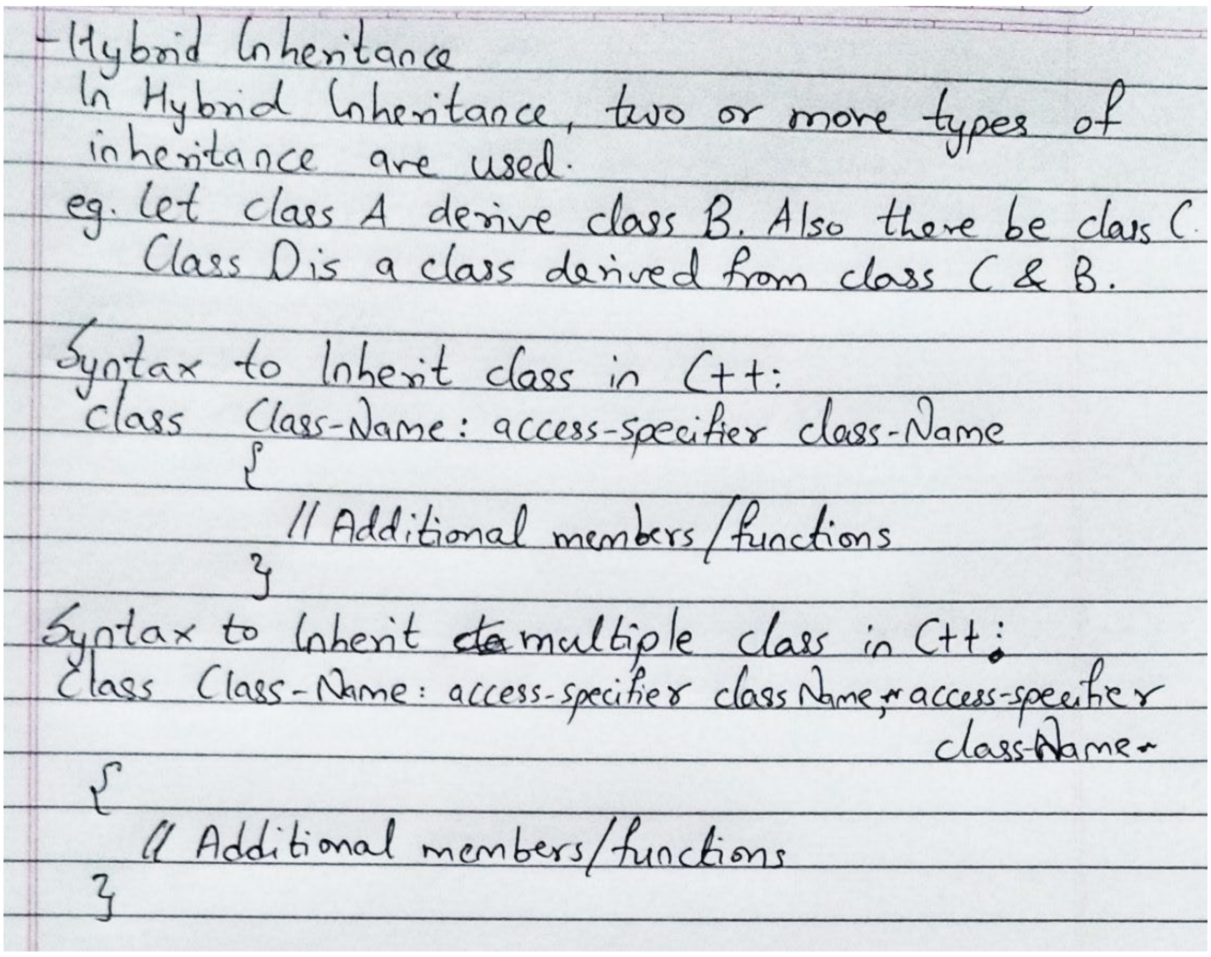
Name: Om Kshirsagar Subject: OOPS
Class: FYIT Date: 2/3/2021



- Hierarchical Inheritance:

In Hierarchical Inheritance, a base class is used to support to more that One class.

eg. Let A be a parent class to class B & class C.



- Hybrid Inheritance

In Hybrid Inheritance, two or more types of inheritance are used.

eg. let class A derive class B. Also there be class C. Class D is a class derived from class C & B.

Syntax to Inherit class in C++:

```
class  Class-Name: access-specifier class-Name
         {
            // Additional members/functions
         }
```

Syntax to Inherit ~~cla~~ multiple class in C++:

```
Class  Class-Name: access-specifier class Name, access-specifier
                                                   class-Name
  {
     // Additional members/functions
  }
```