# Assignment No-3

**Programming Laboratory-I**

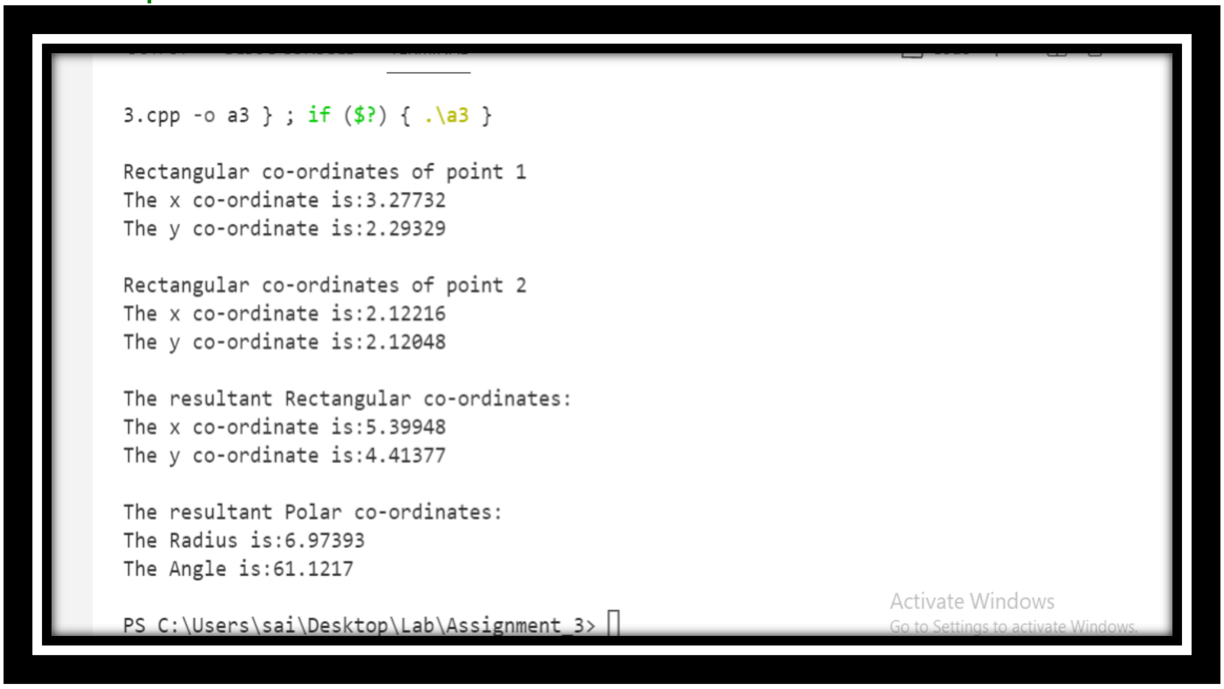
**(Inheritance and Operator Overloading)**

**2020BTECS00005**

**Jadhav Sanket Shivaji.**

```
 (1) .Design Polar Class
#include <iostream>
#include <cmath>
#define PI 3.14
using namespace std;

class polar
{
private:
    float r;
    float a;
    float x;
    float y;
public:
    polar()
    {
        r = 0;
        a = 0;
        x = 0;
        y = 0;
    }
    polar(int radius, int angle)
    {
        r = radius;
        a = angle;
        x = 0;
        y = 0;
    }
    void ConversionToRect(polar &s)
    {
        s.x = (s.r) * cos(s.a * PI / 180.0);
        s.y = (s.r) * sin(s.a * PI / 180.0);
    }
    void ConversionToPolar(polar &p)
    {
        p.r = sqrt(x * x + y * y);
        p.a = (180 / PI) * tan(y / x);
    }
    polar operator+(polar &s)
    {
        polar temp;
        temp.x = x + s.x;
        temp.y = y + s.y;
        return temp;
   }
    void DisplayInRect()
    {
        cout << "The x co-ordinate is:" << x <<
endl;
        cout << "The y co-ordinate is:" << y <<
endl;
    }
    void DisplayInPolar()
    {
        cout << "The Radius is:" << r << endl;
        cout << "The Angle is:" << a << endl;
    }
};
int main()
{
    polar c1(4, 35), c2(3, 45), c3;
    cout<<endl;
    c1.ConversionToRect(c1);
    c2.ConversionToRect(c2);
    cout << "Rectangular co-ordinates of point 1"
<< endl;
```

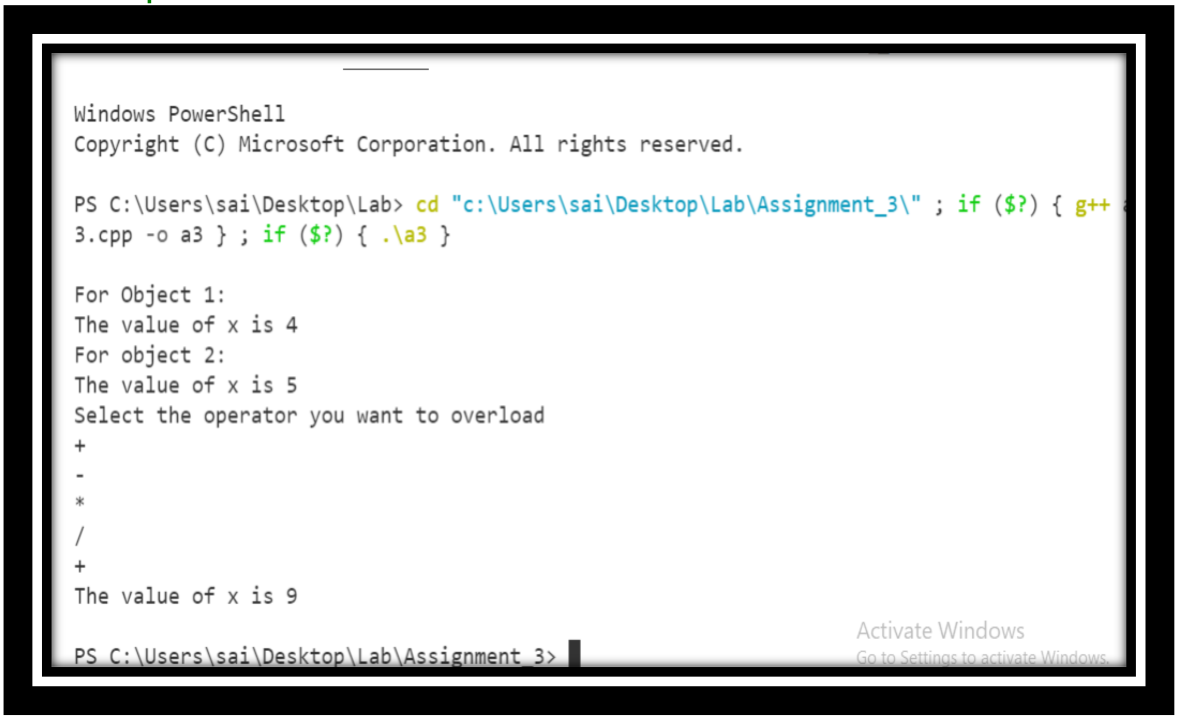
```
    c1.DisplayInRect();
    cout << endl;
    cout << "Rectangular co-ordinates of point 2"
<< endl;
    c2.DisplayInRect();
    cout << endl;
    //Operator Overloading
    c3 = c1 + c2;
    cout << "The resultant Rectangular co-
ordinates:" << endl;
    c3.DisplayInRect();
    cout << endl;
    c3.ConversionToPolar(c3);
    cout << "The resultant Polar co-ordinates:" <<
endl;
    c3.DisplayInPolar();
   cout<<endl;
 return 0;
}
// Output:
```



```
3.cpp -o a3 } ; if ($?) { .\a3 }

Rectangular co-ordinates of point 1
The x co-ordinate is:3.27732
The y co-ordinate is:2.29329

Rectangular co-ordinates of point 2
The x co-ordinate is:2.12216
The y co-ordinate is:2.12048

The resultant Rectangular co-ordinates:
The x co-ordinate is:5.39948
The y co-ordinate is:4.41377

The resultant Polar co-ordinates:
The Radius is:6.97393
The Angle is:61.1217

PS C:\Users\sai\Desktop\Lab\Assignment_3>
```

```
(2) .Create Class Float
#include <iostream>
```

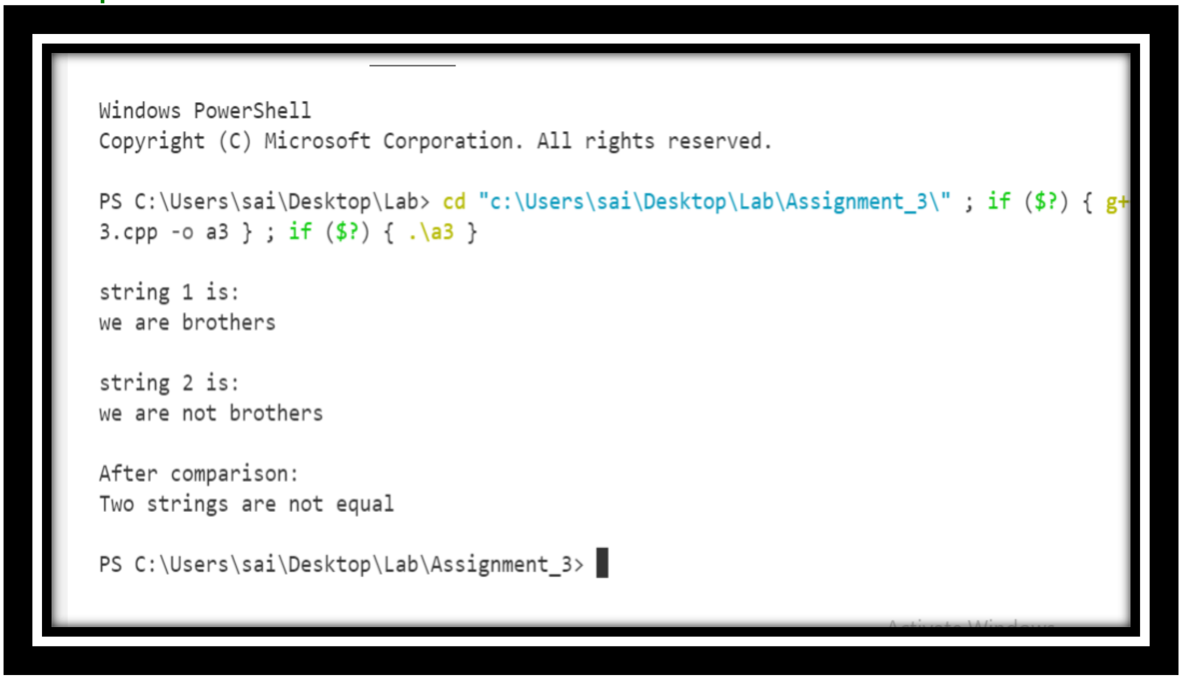
```
using namespace std;
class FLOAT
{
private:
    float x;
public:
    FLOAT()
    {}
    FLOAT(int a)
    {
        x = a;
  }
    FLOAT operator+(FLOAT &s) //Operator
overloading for '+'
    {
        FLOAT temp;
        temp.x = x + s.x;
        return temp;
    }
    FLOAT operator-(FLOAT &s) //Operator
overloading for '-'
    {
        FLOAT temp;
        temp.x = x - s.x;
        return temp;
    }
    FLOAT operator*(FLOAT &s) //Operator
overloading for '*'
    {
        FLOAT temp;
        temp.x = x * s.x;
        return temp;
    }
    FLOAT operator/(FLOAT &s) //Operator
overloading for '/'
    {
        FLOAT temp;
        temp.x = x / s.x;
        return temp;
    }
    void display()
    {

        cout << "The value of x is " << x << endl;
    }
};

int main()
{
    FLOAT f1(4), f2(5), f3;
    cout<<endl;
    cout << "For Object 1:" << endl;
    f1.display();
    cout << "For object 2:" << endl;
    f2.display();
   char ch;
    cout << "Select the operator you want to
overload" << endl;
    cout << "+" << endl;
    cout << "-" << endl;
    cout << "*" << endl;
    cout << "/" << endl;
    cin >> ch;
    switch (ch)
    {
    case '+':
        f3 = f1 + f2;
        f3.display();
        break;
    case '-':
        f3 = f1 - f2;
        f3.display();
        break;
    case '*':
        f3 = f1 * f2;
        f3.display();
        break;
    case '/':
        f3 = f1 / f2;
        f3.display();
        break;
    default:
        cout << "Wrong selection" << endl;
    }
    cout<<endl;
    return 0;
}
// Output:
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sai\Desktop\Lab> cd "c:\Users\sai\Desktop\Lab\Assignment_3\" ; if ($?) { g++
3.cpp -o a3 } ; if ($?) { .\a3 }

For Object 1:
The value of x is 4
For object 2:
The value of x is 5
Select the operator you want to overload
+
-
*
/
+
The value of x is 9

PS C:\Users\sai\Desktop\Lab\Assignment_3>
```

```
(3)Create class string
#include <iostream>
#include <string.h>
using namespace std;
class String
{
private:
    string str;
public:
    String()
    {
    }
    String(string s)
    {
        str = s;
    }
    void display()
    {
        cout << str << endl;
    }
    int operator==(String &a)
    {
        if ((str == (a.str)))
        {
            return 1;
        }
        else
        {
            return 0;
        }
    }
};

int main()
{
    String s1("we are brothers"), s2("we are not
brothers");
    cout << endl;
    cout << "string 1 is:" << endl;
    s1.display();
    cout << endl;
    cout << "string 2 is:" << endl;
    s2.display();
    cout << endl;
    cout << "After comparison:" << endl;
    if ((s1 == s2))
    {
        cout << "Two strings are equal" << endl;
    }
    else
    {
        cout << "Two strings are not equal" <<
endl;
    }
    cout << endl;
    return 0;
}
```

Output:

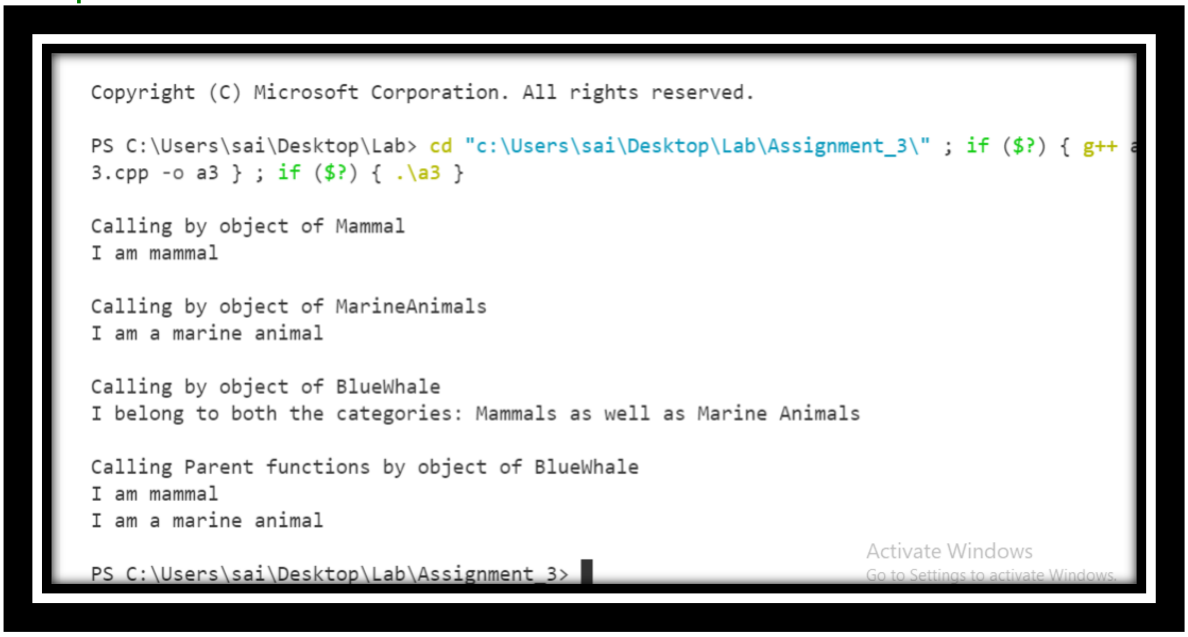


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sai\Desktop\Lab> cd "c:\Users\sai\Desktop\Lab\Assignment_3\" ; if ($?) { g+
3.cpp -o a3 } ; if ($?) { .\a3 }

string 1 is:
we are brothers

string 2 is:
we are not brothers

After comparison:
Two strings are not equal

PS C:\Users\sai\Desktop\Lab\Assignment_3>
```

(5)Mammals Class

```
#include <iostream>
using namespace std;
class Mammals
{
public:
    void displayMammals()
    {
        cout << "I am mammal" << endl;
    }
};
class MarineAnimals
{
public:
    void displayMarineAnimals()
    {
        cout << "I am a marine animal" << endl;
    }
};
class BlueWhale : public Mammals, public
MarineAnimals
{
public:
    void displayBlueWhale()
    {
        cout << "I belong to both the categories:
Mammals as well as Marine Animals" << endl;
    }
};
int main()
{
    Mammals dog;
    MarineAnimals frog;
    BlueWhale whale;
    cout << endl;
    cout << "Calling by object of Mammal" << endl;
    dog.displayMammals();
    cout << endl;
    cout << "Calling by object of MarineAnimals" <<
endl;
    frog.displayMarineAnimals();
    cout << endl;
    cout << "Calling by object of BlueWhale" <<
endl;
    whale.displayBlueWhale();
    cout << endl;
    cout << "Calling Parent functions by object of
BlueWhale" << endl;
    whale.displayMammals();
    whale.displayMarineAnimals();
    cout << endl;
   return 0;
}
```

Output:



```
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sai\Desktop\Lab> cd "c:\Users\sai\Desktop\Lab\Assignment_3\" ; if ($?) { g++ a
3.cpp -o a3 } ; if ($?) { .\a3 }

Calling by object of Mammal
I am mammal

Calling by object of MarineAnimals
I am a marine animal

Calling by object of BlueWhale
I belong to both the categories: Mammals as well as Marine Animals

Calling Parent functions by object of BlueWhale
I am mammal
I am a marine animal

PS C:\Users\sai\Desktop\Lab\Assignment_3>
```

## (6)Vehicles Class

```
#include <iostream>
using namespace std;
class Vehicle
{
public:
    float mileage;
    double price;
};
class Car : public Vehicle
{
protected:
    double owncost;
    int warranty;
    int seatcapacity;
    string fueltype;
};
class Bike : public Vehicle
{
protected:
    int cylinders;
    int gears;
    string coolingtype;
    string wheeltype;
    float tanksize;
};
class Audi : private Car
{
private:
    string model;

public:
    Audi()
    {
    }
    void getCarDetails(string s, double cost, int
warr, int seat, string fuel, float mile, double x)
    {
        model = s;
        owncost = cost;
        warranty = warr;
        seatcapacity = seat;
        fueltype = fuel;
        mileage = mile;
        price = x;
    }
    void displayCarInfo()
    {
        cout << "Model of the car is:  " << model
<< endl;
        cout << "OwnershipCost of the car is:  " <<
std::fixed << owncost << endl;
```

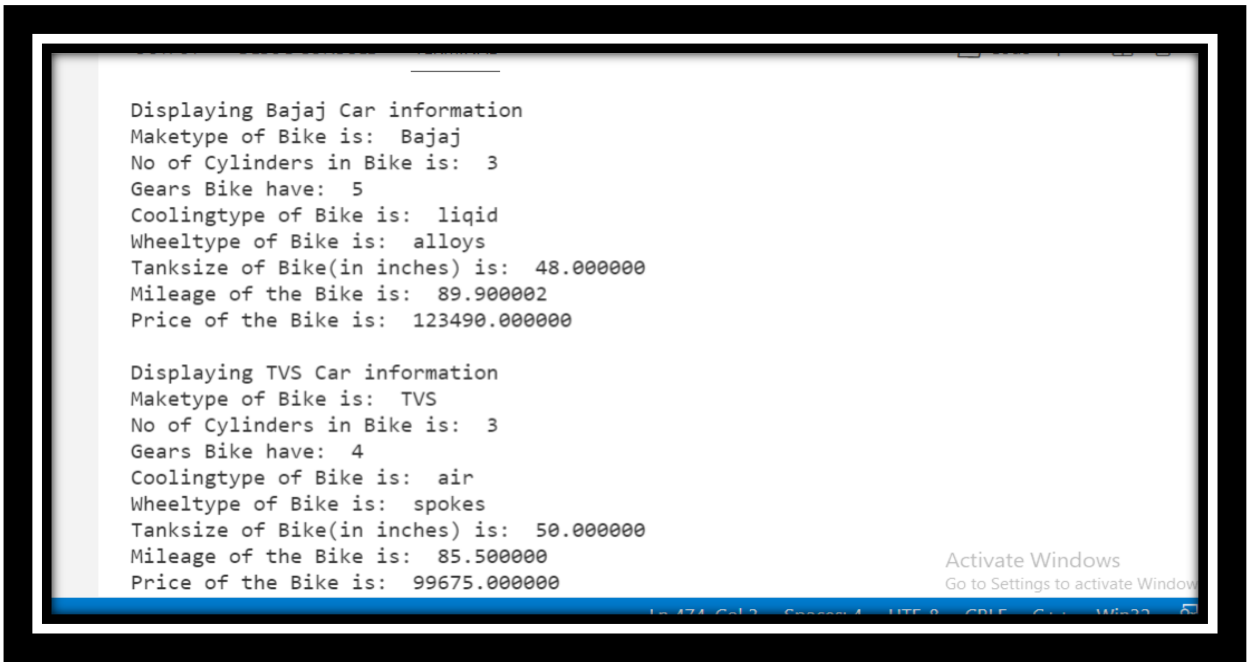
```
        cout << "warranty of the car is:  " <<
warranty << endl;
        cout << "SeatCapacity of the car is:  " <<
seatcapacity << endl;
        cout << "Fueltype of the car is:  " <<
fueltype << endl;
        cout << "Mileage of the car is:  " <<
mileage << endl;
        cout << "price of the car is:  " <<
std::fixed << price << endl;
    }
};
class Ford : private Car
{
private:
    string model;

public:
    void getCarDetails(string s, double cost, int
warr, int seat, string fuel, float mile, double x)
    {
        model = s;
        owncost = cost;
        warranty = warr;
        seatcapacity = seat;
        fueltype = fuel;
        mileage = mile;
        price = x;
    }
    void displayCarInfo()
    {
        cout << "Model of the car is:  " << model
<< endl;
        cout << "OwnershipCost of the car is:  " <<
std::fixed << owncost << endl;
        cout << "warranty of the car is:  " <<
warranty << endl;
```

```
        cout << "SeatCapacity of the car is:  " <<
seatcapacity << endl;
        cout << "Fueltype of the car is:  " <<
fueltype << endl;
        cout << "Mileage of the car is:  " <<
mileage << endl;
        cout << "price of the car is:  " <<
std::fixed << price << endl;
    }
};
class Bajaj : private Bike
{
private:
    string maketype;
public:
    void getBikeDetails(string s, int cylin, int
gear, string cool, string wheel, float size, float
mile, double x)
    {
        maketype = s;
        cylinders = cylin;
        gears = gear;
        coolingtype = cool;
        wheeltype = wheel;
        tanksize = size;
        mileage = mile;
        price = x;
    }
    void displayBikeInfo()
    {
        cout << "Maketype of Bike is:  " <<
maketype << endl;
        cout << "No of Cylinders in Bike is:  " <<
cylinders << endl;
        cout << "Gears Bike have:  " << gears <<
endl;
```

```
        cout << "Coolingtype of Bike is:  " <<
coolingtype << endl;
        cout << "Wheeltype of Bike is:  " <<
wheeltype << endl;
        cout << "Tanksize of Bike(in inches) is:  "
<< tanksize << endl;
        cout << "Mileage of the Bike is:  " <<
mileage << endl;
        cout << "Price of the Bike is:  " <<
std::fixed << price << endl;
    }
};
class TVS : private Bike
{
private:
    string maketype;

public:
    void getBikeDetails(string s, int cylin, int
gear, string cool, string wheel, float size, float
mile, double x)
    {
        maketype = s;
        cylinders = cylin;
        gears = gear;
        coolingtype = cool;
        wheeltype = wheel;
        tanksize = size;
        mileage = mile;
        price = x;
    }
    void displayBikeInfo()
    {
        cout << "Maketype of Bike is:  " <<
maketype << endl;
        cout << "No of Cylinders in Bike is:  " <<
cylinders << endl;
```

```
        cout << "Gears Bike have:  " << gears << 
endl;
        cout << "Coolingtype of Bike is:  " << 
coolingtype << endl;
        cout << "Wheeltype of Bike is:  " << 
wheeltype << endl;
        cout << "Tanksize of Bike(in inches) is:  " 
<< tanksize << endl;
        cout << "Mileage of the Bike is:  " << 
mileage << endl;
        cout << "Price of the Bike is:  " << 
std::fixed << price << endl;
    }
};

int main()
{
    Audi car1;
    Ford car2;
   Bajaj bike1;
    TVS bike2;
    cout << endl;
    car1.getCarDetails("Audi A4", 1123455, 5, 6, 
"Diesel", 34.55, 1278955);
    cout<<"Displaying Audi Car information"<<endl;
    cout<<endl;
    car1.displayCarInfo();
    cout << endl;
     car2.getCarDetails("Aspire", 1234678, 4, 5, 
"Petrol", 23.34, 1345678);

     cout<<"Displaying Ford Car information"<<endl;
     cout<<endl;
     car2.displayCarInfo();

    cout << endl;
    bike1.getBikeDetails("Bajaj", 3, 5, "liqid", 
"alloys", 48, 89.90, 123490);
    cout<<"Displaying Bajaj Car information"<<endl;
    bike1.displayBikeInfo();


    cout << endl;
    bike2.getBikeDetails("TVS", 3, 4, "air", 
"spokes", 50, 85.50, 99675);
    cout<<"Displaying TVS Car information"<<endl;
    bike2.displayBikeInfo();
    cout << endl;
    return 0;
}
```

Output:



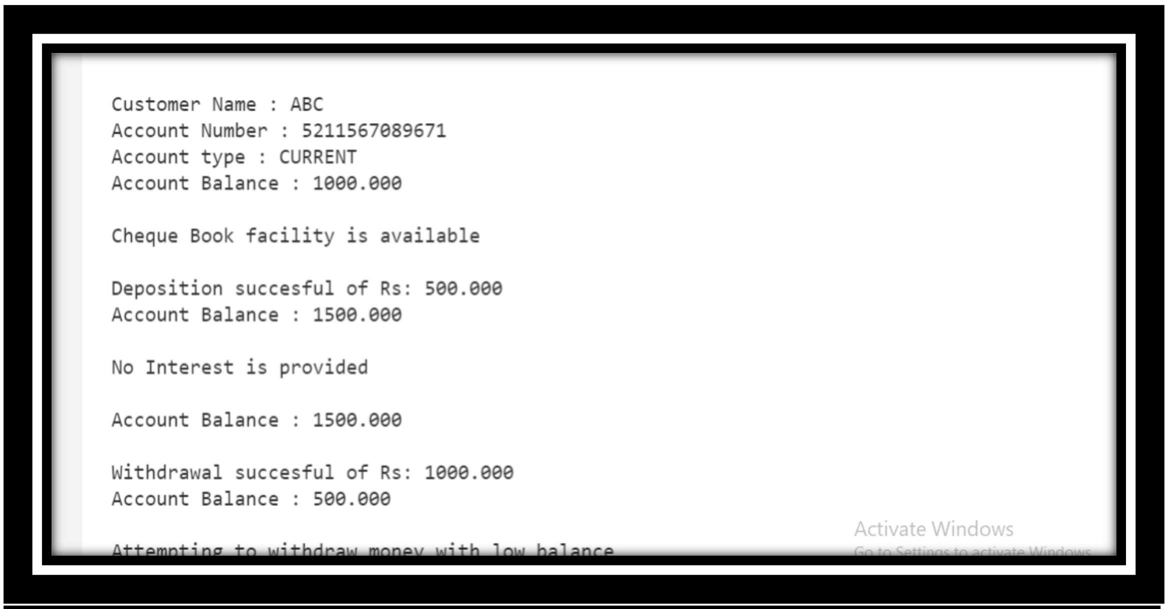
```
Displaying Bajaj Car information
Maketype of Bike is:  Bajaj
No of Cylinders in Bike is:  3
Gears Bike have:  5
Coolingtype of Bike is:  liqid
Wheeltype of Bike is:  alloys
Tanksize of Bike(in inches) is:  48.000000
Mileage of the Bike is:  89.900002
Price of the Bike is:  123490.000000

Displaying TVS Car information
Maketype of Bike is:  TVS
No of Cylinders in Bike is:  3
Gears Bike have:  4
Coolingtype of Bike is:  air
Wheeltype of Bike is:  spokes
Tanksize of Bike(in inches) is:  50.000000
Mileage of the Bike is:  85.500000
Price of the Bike is:  99675.000000
```

(7).Bank CLass

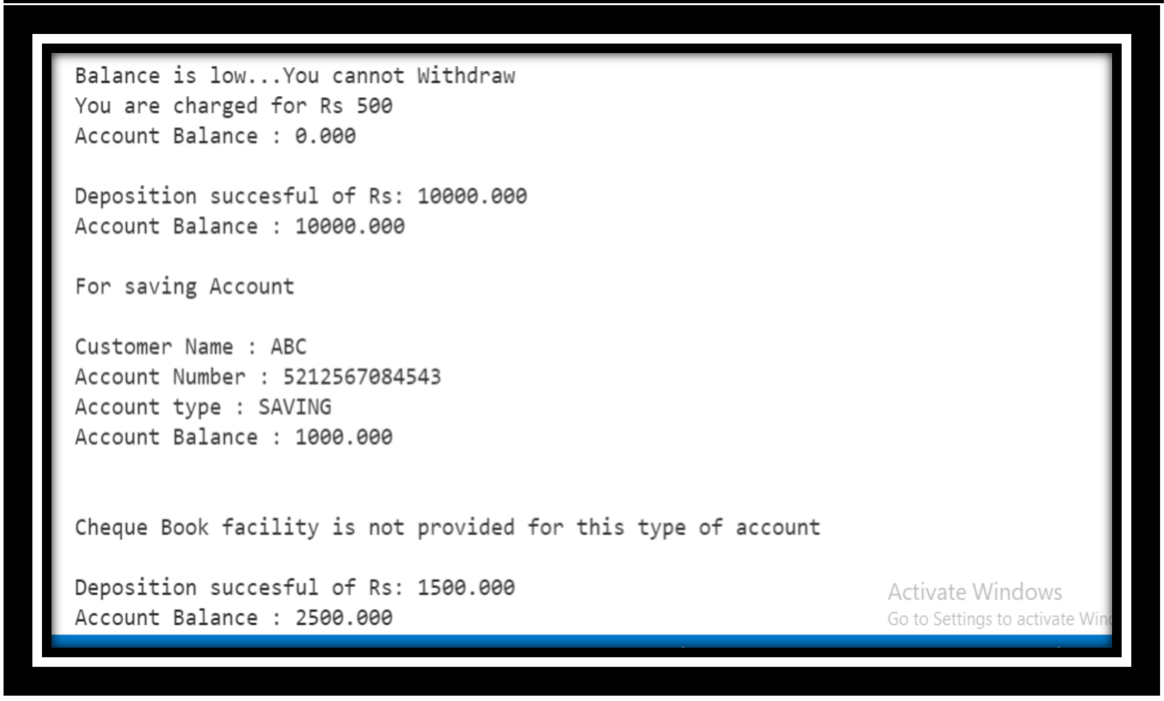
```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
class Account
{
protected:
    string cus_name;
    long double acc_number;
    string acc_type;
};
class cur_acc : private Account
{
private:
    float balance;
    const float minimum_balance = 1000;

public:
    cur_acc()
    {
    }
    void createCustomer(string s, double number,
string p, float initial)
    {
        cus_name = s;
        acc_number = number;
        acc_type = p;
        balance = initial;
   }

   void displayCustmorInfo()
    {
        cout << endl;
        cout << "Customer Name : " << cus_name <<
endl;
        cout << fixed << setprecision(0) <<
"Account Number : " << acc_number << endl;
        cout << "Account type : " << acc_type <<
endl;
        cout << fixed << setprecision(3) <<
"Account Balance : " << balance << endl;
        cout << endl;
    }
    void applyChequeBook()
    {
        cout << "Cheque Book facility is available"
<< endl;
        cout << endl;
    }
    void computeInterest(float r, int n, int t)
    {
        cout << "No Interest is provided" << endl;
        cout << endl;
    }
    void deposit(float depos)
    {
        balance = balance + depos;
        cout << "Deposition succesful of Rs: " <<
depos << endl;
        displayBalance();
    }

    int checkMinimumBal()
    {
        if (balance > minimum_balance)
        {
            return 1;
        }
        else
        {

            return 0;
        }
    }

    void withdrawal(float a)
    {
        if (checkMinimumBal())
        {
            balance = balance - a;
            cout << "Withdrawal succesful of Rs: "
<< a << endl;
```

```
            displayBalance();
        }
        else
        {
            cout << "Balance is low...You cannot
Withdraw" << endl;
            cout << "You are charged for Rs 500" <<
endl;
            balance = balance - 500.000;
            displayBalance();
        }
    }

    void checkBalance()
    {
        displayBalance();
    }

    void displayBalance()
    {

        cout << "Account Balance : " << balance <<
endl;
        cout << endl;
    }
};
class sav_acc : private Account
{
private:
    float balance;
    float minimum_balance = 1000;

public:
    sav_acc(){

    }
    void createCustomer(string s, double number,
string p, float initial)
    {
        cus_name = s;
        acc_number = number;
        acc_type = p;
        balance = initial;
    }
    void displayCustmorInfo()
    {
        cout << endl;
        cout << "Customer Name : " << cus_name <<
endl;
        cout << fixed << setprecision(0) <<
"Account Number : " << acc_number << endl;
        cout << "Account type : " << acc_type <<
endl;
        cout << fixed << setprecision(3) <<
"Account Balance : " << balance << endl;
        cout << endl;
    }
    void applyChequeBook()
   {
        cout << endl;
        cout << "Cheque Book facility is not
provided for this type of account" << endl;
        cout << endl;
    }
    void computeInterest(float r, int n, int t)
    {
        float final_amount = (balance * pow((1 + r
/ 100), n * t)) - balance;
        cout << "Compound Interest :  " <<
final_amount << endl;
        balance = balance + final_amount;
        cout << "Interest is deposited" << endl;
        displayBalance();
    }

    void deposit(float depos)
    {
        balance = balance + depos;
        cout << "Deposition succesful of Rs: " <<
depos << endl;
        displayBalance();
    }

    int checkMinimumBal()
    {
        if (balance > minimum_balance)
        {
            return 1;
        }
        else
        {

            return 0;
        }
    }
   void withdrawal(float a)
    {
        if (checkMinimumBal())
        {
            balance = balance - a;
            cout << "Withdrawal succesful of Rs: "
<< a << endl;
            displayBalance();
        }
        else
        {
            cout << "Balance is low...You cannot
Withdraw" << endl;
            cout << "You have penalty of 500" <<
endl;
```

```
            balance = balance - 500.000;
            displayBalance();
        }
    }

    void checkBalance()
    {
        displayBalance();
   }

   void displayBalance()
    {
        cout << "Account Balance : " << balance <<
endl;
        cout << endl;
    }
};
int main()
{
    // Implementing various functionality for
current account
     cout << "For current Account" << endl;
     cur_acc c1;
     c1.createCustomer("ABC", 5211567089671,
"CURRENT", 1000.00);
     c1.displayCustmorInfo();
     c1.applyChequeBook;
     c1.deposit(500.00);
     c1.computeInterest(2, 5, 3);
     c1.checkBalance;
     c1.withdrawal(1000);
     cout<<"Attempting to withdraw money with low
balance"<<endl;
     c1.withdrawal(1000);
     c1.deposit(10000);

    cout << "For saving Account" << endl;
    sav_acc c2;
    c2.createCustomer("ABC",5212567084543,"SAVING",
1000.00);
    c2.displayCustmorInfo();
    c2.applyChequeBook();
    c2.deposit(1500.00);
    c2.computeInterest(2,3,5);
    c2.checkBalance();
    c2.withdrawal(1000);

    return 0;
}
Output:
```



```
Customer Name : ABC
Account Number : 5211567089671
Account type : CURRENT
Account Balance : 1000.000

Cheque Book facility is available

Deposition succesful of Rs: 500.000
Account Balance : 1500.000

No Interest is provided

Account Balance : 1500.000

Withdrawal succesful of Rs: 1000.000
Account Balance : 500.000

Attempting to withdraw money with low balance
```



```
Balance is low...You cannot Withdraw
You are charged for Rs 500
Account Balance : 0.000

Deposition succesful of Rs: 10000.000
Account Balance : 10000.000

For saving Account

Customer Name : ABC
Account Number : 5212567084543
Account type : SAVING
Account Balance : 1000.000


Cheque Book facility is not provided for this type of account

Deposition succesful of Rs: 1500.000
Account Balance : 2500.000
```

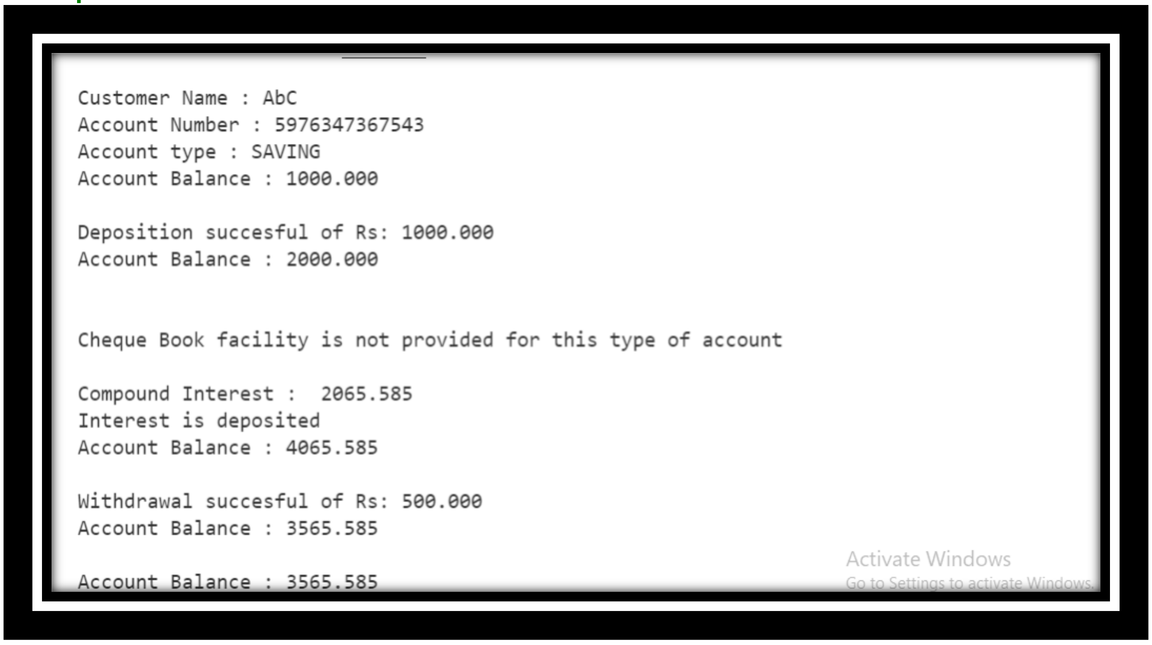
(8)Modify Above Code

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

class Account
{
```

```
protected:
    string cus_name;
    long double acc_number;
    string acc_type;
    public:
    Account()
    {

    }
    Account(string s,long double number,string p)
    {
        cus_name=s;
        acc_number=number;
        acc_type=p;
    }
};
class cur_acc : private Account
{
private:
    float balance;
    const float minimum_balance = 1000;

public:
    cur_acc()
    {
    }
    cur_acc(string s,long double number, string p,
float initial):Account(s,number,p)
    {
        balance = initial;
    }

    void displayCustmorInfo()
    {
        cout << endl;
        cout << "Customer Name : " << cus_name <<
endl;
```

```
        cout << fixed << setprecision(0) <<
"Account Number : " << acc_number << endl;
        cout << "Account type : " << acc_type <<
endl;
        cout << fixed << setprecision(3) <<
"Account Balance : " << balance << endl;
        cout << endl;
    }

    void applyChequeBook()
    {
        cout << "Cheque Book facility is available"
<< endl;
        cout << endl;
    }

    void computeInterest(float r, int n, int t)
    {
        cout << "No Interest is provided" << endl;
        cout << endl;
    }

    void deposit(float depos)
    {
        balance = balance + depos;
        cout << "Deposition succesful of Rs: " <<
depos << endl;
        displayBalance();
    }

    int checkMinimumBal()
    {
        if (balance > minimum_balance)
        {
            return 1;
        }
        else
        {

            return 0;
        }
    }

    void withdrawal(float a)
    {
        if (checkMinimumBal())
        {
            balance = balance - a;
            cout << "Withdrawal succesful of Rs: "
<< a << endl;
            displayBalance();
        }
        else
        {
            cout << "Balance is low...You cannot
Withdraw" << endl;
            cout << "You are charged for Rs 500" <<
endl;
            balance = balance - 500.000;
            displayBalance();
        }
    }

    void checkBalance()
    {
        displayBalance();
    }

    void displayBalance()
    {

        cout << "Account Balance : " << balance <<
endl;
        cout << endl;
    }
};
class sav_acc : private Account
{
private:
    float balance;
    float minimum_balance = 1000;

public:
    sav_acc(string s, double number, string p,
float initial):Account(s,number,p)
    {
        balance = initial;
    }
    void displayCustmorInfo()
    {
        cout << endl;
        cout << "Customer Name : " << cus_name <<
endl;
        cout << fixed << setprecision(0) <<
"Account Number : " << acc_number << endl;
        cout << "Account type : " << acc_type <<
endl;
        cout << fixed << setprecision(3) <<
"Account Balance : " << balance << endl;
        cout << endl;
    }
    void applyChequeBook()
    {
        cout << endl;
        cout << "Cheque Book facility is not
provided for this type of account" << endl;
        cout << endl;
    }
    void computeInterest(float r, int n, int t)
    {
        float final_amount = (balance * pow((1 + r
/ 100), n * t)) - balance;
        cout << "Compound Interest :  " <<
final_amount << endl;
        balance = balance + final_amount;
        cout << "Interest is deposited" << endl;
        displayBalance();
    }

    void deposit(float depos)
    {
        balance = balance + depos;
        cout << "Deposition succesful of Rs: " <<
depos << endl;
        displayBalance();
    }

    int checkMinimumBal()
    {
        if (balance > minimum_balance)
        {
            return 1;
        }
        else
        {

            return 0;
        }
    }
    void withdrawal(float a)
    {
        if (checkMinimumBal())
        {
            balance = balance - a;
            cout << "Withdrawal succesful of Rs: "
<< a << endl;
            displayBalance();
        }
        else
        {
            cout << "Balance is low...You cannot 
Withdraw" << endl;
            cout << "You have penalty of 500" << 
endl;
            balance = balance - 500.000;
            displayBalance();
        }
    }

    void checkBalance()
    {
        displayBalance();
    }

    void displayBalance()
    {
        cout << "Account Balance : " << balance << 
endl;
        cout << endl;
    }
};

int main()
{

     cur_acc 
c1("ABC",5957576456742,"CURRENT",1000.00);
     c1.displayCustmorInfo();
     c1.deposit(1000);
     c1.applyChequeBook();
     c1.computeInterest(3,4,6);
     c1.withdrawal(500);
    c1.displayBalance();
    // Implementing various functionality for
saving account
    sav_acc c2("AbC",5976347367543,"SAVING",1000);
    c2.displayCustmorInfo();
    c2.deposit(1000);
    c2.applyChequeBook();
    c2.computeInterest(3,4,6);
    c2.withdrawal(500);
    c2.displayBalance();
    return 0;
}
```

Output:



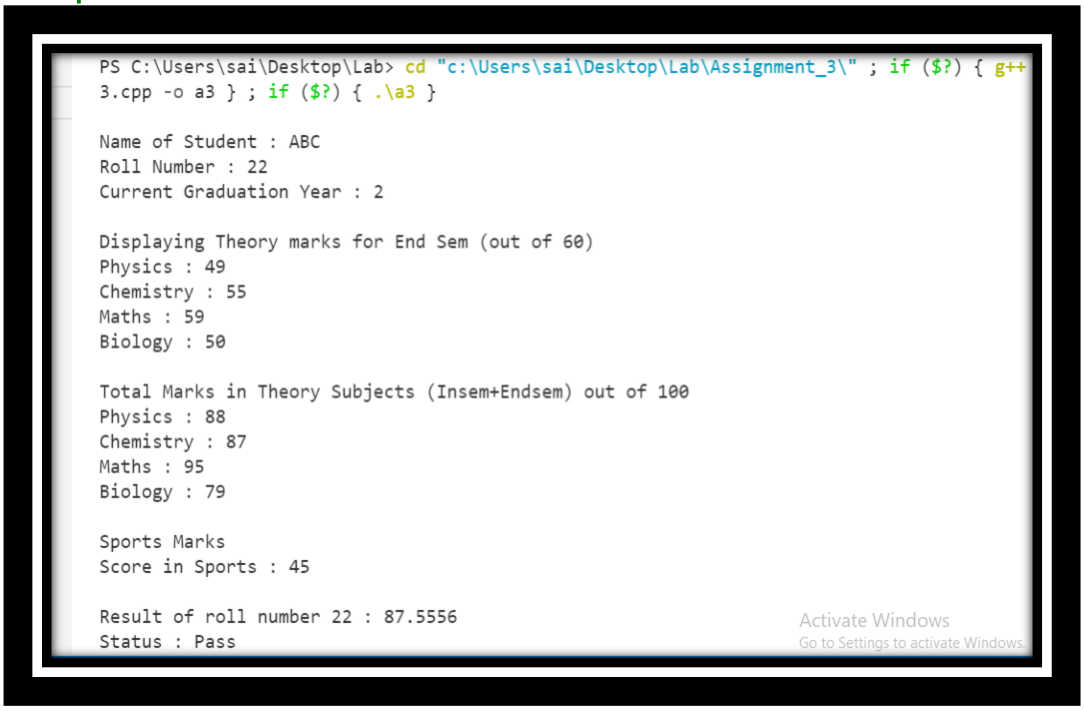
```
Customer Name : AbC
Account Number : 5976347367543
Account type : SAVING
Account Balance : 1000.000

Deposition succesful of Rs: 1000.000
Account Balance : 2000.000


Cheque Book facility is not provided for this type of account

Compound Interest :  2065.585
Interest is deposited
Account Balance : 4065.585

Withdrawal succesful of Rs: 500.000
Account Balance : 3565.585

Account Balance : 3565.585
```

(9).Hybrid Inheritance

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

class Student
{

public:
    string name;
    int roll_number;
    int graduation_year;
    Student()
    {

    }
    Student(string s,int x, int y)
    {
        name=s;
        roll_number = x;
        graduation_year = y;
    }
    void StudentInfo()
    {
        cout<<endl;
        cout << "Name of Student : " << name <<
endl;
        cout << "Roll Number : " << roll_number <<
endl;
        cout << "Current Graduation Year : " <<
graduation_year << endl;
    }
};

class InSemScore : virtual public Student
{
protected:
    float insemphy;
    float insemchem;
    float insemmath;
    float insembio;

public:
    InSemScore()
    {

    }
    InSemScore(float a, float b, float c, float d)
    {
        insemphy = a;
        insemchem = b;
        insemmath = c;
        insembio = d;
    }
    void displayInSemMarks()
    {
        cout << endl;
        cout << "Displaying Theory marks for In Sem
(out of 40)" << endl;
        cout << "Physics : " << insemphy << endl;
        cout << "Chemistry : " << insemchem <<
endl;
        cout << "Maths : " << insemmath << endl;
        cout << "Biology : " << insembio << endl;
    }
};
class EndSemScore : virtual public Student
{
protected:
    float endsemphy;
    float endsemchem;
    float endsemmath;
    float endsembio;

public:
   EndSemScore()
    {

    }
    EndSemScore(float e, float f, float g, float h)
    {
        endsemphy = e;
        endsemchem = f;
        endsemmath = g;
        endsembio = h;
    }
    void displayEndSemMarks()
    {
        cout << endl;
        cout << "Displaying Theory marks for End
Sem (out of 60)" << endl;
        cout << "Physics : " << endsemphy << endl;
        cout << "Chemistry : " << endsemchem <<
endl;
        cout << "Maths : " << endsemmath << endl;
        cout << "Biology : " << endsembio << endl;
    }
};
class Sports : virtual public Student
{
protected:
    float score;

public:
    Sports()
    {

    }
    Sports(float i)
    {
        score = i;
    }
    void displaySportsScore()
    {
        cout << "Score in Sports is : " << score <<
endl;
    }
};
class Result : public InSemScore, public
EndSemScore, public Sports
{
private:
    float physics;
    float chemistry;
    float maths;
    float biology;
    float result;

public:
    Result(string s,int x, int y, float a, float b,
float c, float d, float e, float f, float g, float
h, float i) : Student(s,x, y), InSemScore(a, b, c,
d), EndSemScore(e, f, g, h), Sports(i)
    {
    }

    void TotalMarks()
    {
        physics = insemphy + endsemphy;
        chemistry = insemchem + endsemchem;
        maths = insemmath + endsemmath;
        biology = insembio + endsembio;
        cout << endl;
        cout << "Total Marks in Theory Subjects
(Insem+Endsem) out of 100" << endl;
        cout << "Physics : " << physics << endl;
        cout << "Chemistry : " << chemistry <<
endl;
        cout << "Maths : " << maths << endl;
        cout << "Biology : " << biology << endl;
        cout << endl;
        cout << "Sports Marks" << endl;
        cout << "Score in Sports : " << score <<
endl;
        cout << endl;
    }
    void PercentageResult()
    {
        result = (((physics + chemistry + maths +
biology + score) / 450) * 100);
```

```
        cout << "Result of roll number " <<
roll_number << " : " << result << endl;
        if (result > 45.00)
        {
            cout << "Status : Pass";
        }
        else
        {
            cout << "Status : Fail";
        }
    }
};
int main()
{
    Result stu1("ABC",22, 2, 39, 32, 36, 29,49, 55,
59, 50, 45);
    stu1.StudentInfo();
    stu1.displayEndSemMarks();
    stu1.TotalMarks();
    stu1.PercentageResult();
    return 0;
}
```

Output:



```
PS C:\Users\sai\Desktop\Lab> cd "c:\Users\sai\Desktop\Lab\Assignment_3\" ; if ($?) { g++
3.cpp -o a3 } ; if ($?) { .\a3 }

Name of Student : ABC
Roll Number : 22
Current Graduation Year : 2

Displaying Theory marks for End Sem (out of 60)
Physics : 49
Chemistry : 55
Maths : 59
Biology : 50

Total Marks in Theory Subjects (Insem+Endsem) out of 100
Physics : 88
Chemistry : 87
Maths : 95
Biology : 79

Sports Marks
Score in Sports : 45

Result of roll number 22 : 87.5556
Status : Pass
```