Virtual base class:

Create a class Student with data member roll\_no and two functions to initialize and display it. Derive two new classes Theory and Practical from Student. Define suitable functions to assign and display theory and practical marks for two different subjects. Again, derive a new class Result from both class Theory and Practical and add a new function to display the final total marks of student. Write a main program to test your class.

#include<iostream>

using namespace std;

class Student

{

private:

int roll;

public:

void setroll()

{

cout<<"Enter roll number";

cin>>roll;

}

void showroll()

{

cout<<"Roll= "<<roll;

}

};

class Theory: public virtual Student

{

protected:

float comth,ength;

public:

void setdatatheory()

{

cout<<"Enter Theory marks of computer and English ";

cin>>comth>>ength;

}

void showmarkstheory()

{

cout<<"Computer(Theory)= "<<comth;

cout<<"English(Theory)= "<<ength;

}

};

class Practical: public virtual Student

{

protected:

float compr,engpr;

public:

void setdatapractical()

{

cout<<"Enter Practical marks of computer and English ";

cin>>compr>>engpr;

}

void showmarkspractical()

{

cout<<"Computer(Practical)= "<<compr;

cout<<"English(Practical)= "<<engpr;

}

};

class Result: public Theory, public Practical

{

public:

void showtotal()

{

float tot;

tot=comth+ength+compr+compr;

cout<<"Total obtained marks= "<<tot;

}

};

int main()

{

Result res;

res.setroll(); //ambiguous because multipath exist to reach setroll() from derived class

//so must use virtual base class to overcome this

res.setdatatheory();

res.setdatapractical();

res.showroll();

res.showmarkstheory();

res.showmarkspractical();

res.showtotal();

}

-----------------------------------------------------

Default Constructor (No arguments) in inheritance

#include<iostream>

using namespace std;

class A { //Base Class

public:

A() { //Base Class A Constructor

cout<<"Constructor Class A";

}

//Base Class A Destructor

~A()

{

cout<<"Destructor Class A";

}

};

class B:public A //Derived class

{

public:

B() //Derived Class B Constructor

{

cout<<"Constructor Class B";

}

~B() { //Derived Class B Destructor

cout<<"Destructor Class B";

}

};

int main()

{

B obj; //Derived class object obj

}

--------------------------------------------------------

Parameterized constructor (with arguments) in inheritance

#include<iostream>

using namespace std;

class A //Base class

{

protected:

int a;

A(int x) //Base class constructor with one argument(x)

{

a=x;

cout<<"Constructor : Class A : value : "<<a<<endl;

}

};

class B:public A //Derived class B from base class A

{

protected:

int b;

public:

//derived class constructor passing arguments(y, z)

//y is used in derived class and z is passed base class A

B(int y,int z):A(z)

{

b=y;

cout<<"Constructor : Class B value : "<<b<<endl;

}

};

int main()

{

//derived class object passing arguments to derived & base class

B obj(5,3);

}

-----------------------------------------------------------------