**Lab-4 Inheritance**

1. Write a program of single inheritance :

Base Class : B

* Private member : a (int)
* Public member: b (int)

Derived Class : D

* Private member : c (int)

Calculate c = a\* b and display

**a) public derivation**

#include<iostream>

using namespace std;

class B

{

int a; //private, not inheritable

public:

int b;

void get\_ab();

int get\_a (void);

void show\_a(void);

};

class D: public B // public derivation

{

int c;

public:

void mul(void);

void display(void);

};

//……………………

void B:: get\_ab(void)

{

a = 5; b = 10;

}

int B:: get\_a()

{

return a;

}

void B:: show\_a()

{

cout<<"a="<<a<<"\n";

}

void D:: mul()

{

c = b \* get\_a(); // a is private can not be inherited

}

void D:: display()

{

cout<<"a="<<get\_a()<<"\n";

cout<<"b="<<b<<"\n";

cout<<"c="<<c<<"\n";

}

int main()

{

D d;

d.get\_ab();

d.mul();

d.show\_a();

d.display();

return 0;

}

**b) private derivation.**

# include <iostream>

using namespace std;

class B

{

int a;

public:

int b;

void get\_ab();

int get\_a(void);

void show\_a(void);

};

class D: private B

{

int c;

public:

void mul (void);

void display (void);

};

// ……………………….

void B:: get\_ab(void)

{

cout<<" Enter value for a and b";

cin>> a >>b;

}

int B:: get\_a()

{

return a;

}

void B:: show\_a ()

{

cout <<"a = " << a <<"\n";

}

void D:: mul ()

{

get\_ab();

c =b \* get\_a(); // a is private

}

void D:: display()

{ show\_a();

cout<<"b="<<"\n";

cout<<"c="<<c<<"\n";

}

int main()

{ D d;

// d.get\_ab(); won't work

d.mul();

// show\_a(); won't work

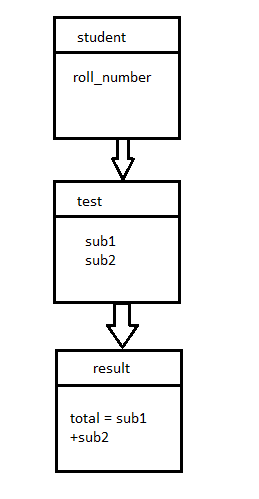
d.display();

//d.b = 20; won't work b is private

return 0;

}

2. Solve the following problem for Multilevel Inheritance:



# include <iostream>

using namespace std;

class student

{

protected:

int roll\_number;

public:

void get\_number (int);

void put\_number (void);

};

void student:: get\_number (int a)

{

roll\_number = a;

}

void student:: put\_number()

{

cout << “roll number:” << roll\_number <<”\n”;

}

class test: public student // first derivation

{

protected:

float sub1;

float sub2;

public:

void get\_marks (float, float);

void put\_marks (void);

};

void test:: get\_marks (float x, float y)

{

sub1 = x;

sub2 = y;

}

void test:: put\_marks ()

{

cout << “marks in sub1 = “<< sub1 <<”\n”;

cout << “marks in sub2 = “<< sub2 <<”\n”;

}

class result: public test // second derivation

{

float total;

public:

void display (void);

};

void result:: display (void)

{

total = sub1 + sub2;

put\_number ();

put\_marks();

cout << “\n total = “ << total;

}

int main ()

{

result student 1;

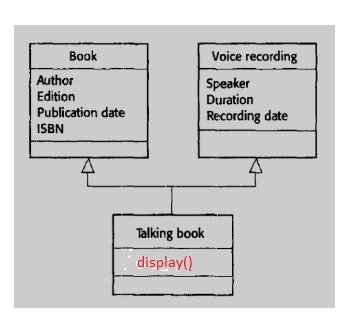
student1.get\_number (111);

student1.get\_marks (75.0, 59.5);

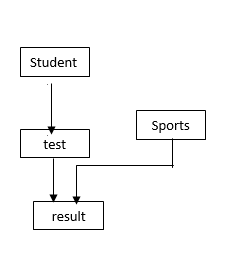
student1.dsiplay();

}

3. Solve the following form of multiple inheritance



4. Solve the following form of Hybrid Inheritance.



class sports

{

protected:

float score;

public:

void get\_score(float);

void put\_score (void)

};

class result: public test, public sports

{

……………….

……………….

……………….

};

class test: public student

{

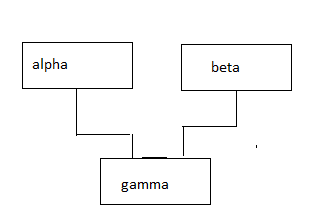
……………….

……………….

……………….

};

5. Solve the following form of inheritance making one constructor in each class.



# include <iostream>

using namespace std;

class alpha

{

int x;

public:

alpha (int i)

{

x = i;

cout <<"alpha initialized \n";

}

void show\_x(void)

{

cout<<"x = "<< x <<"\n";

}

};

class beta

{

float y;

public:

beta (float j)

{

y = j;

cout<<"beta initialized\n";

}

void show\_y (void)

{

cout <<"y="<<y <<"\n";

}

};

class gamma: public beta, public alpha // *order of execution*

{

int m,n;

public:

gamma (int a, float b, int c, int d): alpha (a), beta (b)

{

m = c;

n = d;

cout<<"gamma initialized\n";

}

void show\_mn(void)

{

cout<<"m="<<m<<"\n"

<<"n = " << n <<"\n";

}

};

int main ()

{

gamma g(5, 10.75, 20, 30);

g.show\_x();

g.show\_y();

g.show\_mn();

}

6. Solve Q. No. 5 adding destructor in each class.