**This page must be turned in with the exam Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**// Constructors and Destructors Problem 8 Code**

#include <iostream>

using namespace std;

class A{

 public:

 A(); // Default constructor

 A(int); // int Constructor

 A(const A&); // Copy constructor

 ~A(); // Destructor

 A operator+(const A& rhs) const; // Addition operator

 public:

 int x; // Single data member

};

A::A(): x(0){ cout<<"A Def Con "<<endl; };

A::A(int x): x(x){ cout<<"A Int Con "<<endl; };

A::~A(){ cout<<"-A Destructor "<<endl; };

A::A(const A &a){

 x=a.x;

cout<<"A Copy Con "<<endl;

};

 A A::operator+(const A& rhs) const

{

 cout<<"A + "<<endl;

 A r(x + rhs.x);

return r;

}

class B{

 public:

 B(); // Default Constructor

 B(int); // int Constructor

 B(const B&); // Copy constructor

 ~B(); // Destructor

 B operator+(B rhs) const; // Addition operator

 public:

 int x; // Single data member

};

B::B(): x(0){ cout<< "B Def Con "<<endl;};

B::B(int x): x(x){ cout<< "B Int Con "<<endl;};

B::~B(){ cout<< "-B Destructor "<<endl;};

B::B(const B &b){

 x=b.x;

 cout<<"B Copy Con "<<endl;

};

 B B::operator+(B rhs) const

{

 cout<< "B + "<<endl;

 return B(x + rhs.x);

}

 int main()

{

 A a;

 B b(2);

 B c(b);

 a = a + a;

 b = b + c;

 cout << a.x << " " << b.x <<endl;

}

**// Inheritance and Polymorphism problem 9 code**

#include <iostream>

using namespace std;

class Base

 { // Define a base class

 public:

 virtual void Func1() = 0;

 virtual void Func2();

 virtual void Func3();

 void Func4();

 };

 class A : public Base

 { // Class A derives from Base

 public:

 void Func2();

 void Func4();

 };

 class B : public A

 {// Class B derives from A

 public:

 virtual void Func1();

 void Func2();

 };

 class C : public Base

 { // Class C derives from Base

 public:

 virtual void Func1();

 virtual void Func4();

 };

 // Base Class Methods

 void Base::Func2(){ cout << "Hello from Base::Func2()" << endl;}

 void Base::Func3()

 {

 cout << "Hello from Base::Func3()" << endl;

 Func1(); // DON’T MISS THIS CALL IN YOUR ANSWER

 }

 void Base::Func4(){ cout << "Hello from Base::Func4()" << endl;}

 // Class A Methods

 void A::Func2() { cout << "Hello from A:Func2()" << endl; }

 void A::Func4() { cout << "Hello from A:Func4()" << endl; }

 // Class B Methods

 void B::Func1() { cout << "Hello from B:Func1()" << endl; }

 void B::Func2() { cout << "Hello from B:Func2()" << endl; }

 // Class C Methods

 void C::Func1() { cout << "Hello from C:Func1()" << endl; }

 void C::Func4() { cout << "Hello from C:Func2()" << endl; }

 void TestFunc(Base& x)

 {

 x.Func1();

 x.Func2();

 x.Func3();

 x.Func4();

 }

 int main()

 {

 B b;

 C c;

 TestFunc(b);

 TestFunc(c);

 }