Inheritance Example:

/\* Single Inheritance \*/

#include <iostream>

using namespace std;

class Base

{public:

double salary = 875;

};

class Bonus : public Base

{public:

double bonus = 87.50;

void sum() {

cout << "Your Total Salary is: " << (salary + bonus) << endl;

}

};

int main()

{

Bonus x;

cout << "Your Salary is:" << x.salary << endl;

cout << "Your Bonus is:" << x.bonus << endl;

x.sum();

return 0;

}

Polymorphism 1

/\* Compile Time Polymorphism: Function Overloading \*/

#include <iostream>

#include <string>

using namespace std;

class Addition

{ public: int ADD(int X, int Y)

{

return X + Y;

}

string ADD()

{

string a = "Hello";

string b = "World";

return a + b;

}

};

int main(void)

{

Addition obj;

cout << obj.ADD(20, 30) << endl; // Will add the numbers

cout << obj.ADD()<<endl; // Will concatenate “Hello World”

return 0;

}

/

#include<iostream>

using namespace std;

class Add

{public: virtual void print()

{ int a = 20, b = 30;

cout << " base class Action is:" << a + b << endl;

}

void show()

{ cout << "show base class" << endl; }

};

class Sub : public Add

{public: void print() // print () is already virtual function

// in derived class, we could also declared

// as virtual void print () explicitly

{ int x = 20, y = 10;

cout << " derived class Action:" << x - y << endl;

}

void show()

{ cout << "show derived class" << endl; }

};

int main()

{

Add\* aptr;

Sub s;

aptr = &s;

aptr->print(); // Virtual function, bound at runtime

aptr->show(); // Regular function, bound at compile time

return 0;

}