Name: Jonathan Sum

1. Here is the first line of a class declaration. Circle the name of the base class and underline the name of the derived class:

|  |
| --- |
| Vehicle |

class Truck : public

1. Is the following program legal (assuming appropriate #include and using directives are added)?  
   void show\_employee\_data(const Employee object);

int main( )

{

HourlyEmployee joe(“Mighty Joe”, “123-45-6789”, 20.50, 40);

SalariedEmployee boss(“Mr. Big Shot”, “987-65-4321”, 10500.50);

show\_employee\_data(joe);

show\_employee\_data(boss);

return 0;

}

void show\_employee\_data(const Employee object)

{

cout << “Name: “ << object.get\_name( ) << endl;

cout << “Social Security Number: “ << object.get\_ssn( ) << endl;

}

illegal

1. Give a definition for a class SmartBut that is a derived class of the base class Smart, which we reproduce for you here. Do not bother with #include directives or namespace details.

class Smart

{

public:

Smart( );

void print\_answer( ) const;

protected:

int a;

int b;

};

class SmartBut::public Smart {

public:

bool crazy;

boold is\_crazy();

}

bool SmartBut::is\_crazy(){

return crazy;

}  
This class should have an additional data field, crazy, that is of type *bool*, one additional member function that takes no arguments and returns a value of type *bool*, and suitable constructors. The new function is named is\_crazy. You do not need to give any implementations, just the class definition.

1. Is the following a legal definition of the member function is\_crazy in the derived class SmartBut discussed in exercise 4? Explain your answer. (remember, the question asks if it is legal, not if it is a sensible definition.)  
   bool SmartBut::is\_crazy( ) const

{

if (a > b)

return false;

else

return true;

}

It is legal. The function from the derived class can access the protected varible in the base class.

1. Give a definition for a class TitledEmployee that is a derived class of the base class SalariedEmployee given in lecture notes. The class TitledEmployee has one additional member variable of type string called title. It also has two additional member functions: get\_title, which takes no arguments and returns a string; and set\_title, which is a *void* function that takes one argument of type string. It also redefines the member function set\_name. You do not need to give any implementations, just the class definition. however, do give all needed #include directives and all *using namespace* directives.

class SalariedEmployee{

public:

protected:

string name;

}

#include “SalariedEmployee.h”

using namespace std;

class TitledEmployee: public SalariedEmployee {

public:

string title;

string get\_title() const;

void set\_title(string t);

void set\_name();

}

1. Give the definitions of the constructors for the class TitledEmployee that you gave as the answer to exercise 5. Also, give the redefinition of the member function set\_name. The function set\_name should insert the title into the name. Do not bother with #include directives or namespace details

void TitledEmployee::set\_name(){

name = title;

}

1. You know that an overloaded assignment operator and a copy constructor are not inherited. Does this mean that if you do not define an overloaded assignment operator or a copy constructor for a derived class, then that derived class will have no assignment operator and no copy constructor?

Nope

1. Suppose Child is a class derived from the class Parent, and the class Grandchild is a class derived from the class Child. This question is concerned with the constructors and destructors for the three classes Parent, Child, and Grandchild. When a constructor for the class Grandchild is invoked, what constructors are invoked and in what order? When the destructor for the class Grandchild is invoked, what destructors are invoked and in what order?

The order will be:

1. Parent class’s constructors is invoked.
2. Child class’s constructors is invoked.
3. Grandchild class class’s constructors is invoked.
4. Grandchild class’s destructors is invoked.
5. Child class’s destructors is invoked.
6. Parent class’s destructors is invoked.

Thus, since child class constructors is the last constructors is invoked, destructor for the class Grandchild will be invoked right after that. After the Grandchild destructors is invoked, the Child class destructors will be invoked.