**Pseudo-Exam for CS 121-15, Spring 2015 SI**

**Terms:** Match the definition letter with the below keyword.

A. Pointer to the object whose member function is being executed

B. A class with this specification has access to another class's private/protected members

C. Variables of this type are only created once in a program, even if in a class

D. Used for creating a new object as a copy of an existing object

E. To supply more than one definition for an operator in the same scope

F. Where some code behaves differently in different contexts

G. The idea of bundling data and functions used in a class

H. The idea of only exposing the interfaces and hide implementation details of a class from the user

I. A way to define classes and functions using generic types

J. A class that is inherited from in a class hierarchy

K. A class that inherits in a class hierarchy

L. An instance of a class

M. Method that is declared but has no implementation

N. When an object or class is based on another object or class

\_\_ Data Abstraction \_\_ Operator overloading

\_\_ Derived class \_\_ Encapsulation

\_\_Inheritance \_\_ Object

\_\_ friend \_\_ Abstract Function

\_\_ Polymorphism \_\_ Base class

\_\_ Template \_\_ Static

\_\_ Copy Constructor \_\_ this

**Concept**

1. What are the two parts of any recursive function? Give two examples of where recursion may be used.

2. What is the difference between a class and a structure?

3. What possible issues come from forgetting to close a file after reading/writing from it successfully?

4. In your own words, explain how implementing class inheritance can be considered a useful practice.

5. Note the differences between public, protected, and private inheritance. (NOTE: Specify what gets inherited with each type)

6. Note the difference between a pure virtual method and a virtual method.

**Syntax**

1. Implement a factory function to generate different types of Cars using the below syntax.

class Car {

protected:

int numSeats, numWheels

Car(int seats, int wheels); //assume already implemented

}

class PickupTruck : public Car {

public:

PickupTruck(int seats, int wheels) : Vehicle(seats, wheels)

{ … } //assume implemented

}

class SportsCar : public Car {

public:

SportsCar(int seats, int wheels) : Vehicle(seats, wheels)

{ … } //assume already implemented

}

class Minivan : public Car {

public:

Minivan(int seats, int wheels) : Vehicle(seats, wheels)

{ … } //assume already implemented

}

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

enum CarType { PICKUP = 0, SPORTS, MINIVAN }; //this is an enumerated type for your convenience

Car \* generate\_car(int seats, int numWheels, int type) {

}

2) (Separate of previous class code). Write a function that takes in a vector of characters, copies all uppercase characters to a string called *upper*, all lower case letters to a string called *lower*, and returns *upper* and *lower* concatenated.

3) Spot any errors in the below code.

vector<string> init\_str\_list(string fn) {

fstream file(fn.c\_str());

vector<string> str\_list

if(file.open()) {

string temp;

while( cin >> temp ) {

str\_list.add(temp);

}

}

else { cerr << “Could not open file!\nReturning empty vector.\n”; }

return str\_list;

}