**CMPE 50 – Spring 2021**

**Lab #10 – Inheritance and Polymorphism**

**Instructions**: Before you leave the lab, you should submit your answers through Canvas->Assignment->Lab10->Submission. Please submit your answers (.h and .cpp files, and output files with appropriate documentation/comments) even if you couldn’t complete/run them.

**Exercise 1 – Inheritance**

Write a program that uses the class SalariedEmployee in Display 15.5. The files are available in the files Section (Files->Lab 10 files)

* employee.h
* employee.cpp
* salariedemployee.h
* salariedemployee.cpp

Your program is to define a class called Administrator, which is to be derived from the class SalariedEmployee. You are allowed to change private in the base class to *protected*. You are to supply the following additional data and function members:

A member variable of type string that contains the administrator’s title (such as Director or Vice President)

A member variable of type string that contains the company area of responsibility (such as Productions, Accounting, or Personnel)

A member variable of type string that contains the name of this administrator’s immediate supervisor.

A protected member variable of type double that holds the administrator’s annual salary. It is possible for you to use the existing salary member if you did the change recommended earlier.

A member function called set\_supervisor, which changes the supervisor’s name.

A member function for reading in an administrator’s data from the keyboard.

A member function called print, which outputs the objects data to the screen/file

An overloading of the member function print\_check() with appropriate notations on the check. (Based on programming project 15.1)

**Exercise 2 - Polymorphism**

Consider a graphics system that has classes for various figures – rectangles, squares, triangles, circles, and so on. For example, a rectangle might have data members for height, width, and center point, while square and circle might have only a center point and an edge length or radius, respectively. In a well-designed system, these would be derived from a common class Figure. You are to implement such system.

The class Figure is the base class. You should add Circle and Triangle classes derived from Figure. Each class has stubs for member functions *erase* and *draw*. Each of these member functions outputs a message telling what function has been called and what the class of the calling object is. Since these are just stubs, they do nothing more than output the message. The member function *center* calls the *erase* and *draw* functions to erase and redraw the figure at the center. Since you have only stubs for erase and draw, center will not do any “centering” but will call the member functions erase and draw. Also add an output message in the member function center that announces that center is being called. The member functions take no arguments.

There are three parts to this project:

1. Write the class definitions using no virtual functions. Compile and test.
2. Make the base class member functions virtual. Compile and test.
3. Explain the difference in results.

Use the following main function for all testing: (see next page)

**#include** <iostream>

**#include** "figure.h"

**#include** "circle.h"

**#include** "triangle.h"

**using** **namespace** std;

void myDraw(Figure \*fig)

{

fig->draw();

cout << "\n myDraw: Derived class object calling center(). \n";

fig->center();

}

**int** **main**(){

Figure \*fig;

Triangle \*tri = new Triangle;

fig = tri;

fig->draw();

cout << "\n Derived class Triangle object calling center(). \n";

fig->center();

myDraw(tri);

Circle \*cir = new Circle;

fig = cir;

fig->draw();

cout << "\n Derived class Circle object calling center(). \n";

fig->center();

myDraw(cir);

**return** 0;

}

(based on Prog. Proj. 15.5)