COP3331 Lab 8

Submission Instructions:

1. Create a folder named Lab8_LastName_FirstInitial (e.g. Lab8_Neal_T).
2. In your folder, place a **PDF** file containing your answers to questions with a ♢.
3. Copy your directories containing your programs for questions with a ♠ into the folder; **these directories should only contain files needed to run your program, which may include one or more of the following file types: .cpp, .h., and .txt**. Do NOT include the full project (e.g., solution file). Test your program on CIRCE before submitting by compiling and running with g++. Your file containing main() should **always** be named main.cpp.
4. Ensure that all programs have block comments at the very beginning (starting at the first line) in the file containing main() with your name and the program's description. **The block comment's format should be identical to what's provided in Figure 2-1.**
5. Use single-line comments to describe your code's functionality as needed.
6. Do not submit anything for questions with a ♣.
7. Zip the folder and submit it via Canvas.

♢ = 5 points each, ♠ = 15 points each

---

1. ♣ Read *Chapter 16: More skills for object-oriented programming.*
2. ♢ Given a class named Employee with this data member declaration:
   int hire_year;
   how would you code the setter function so it doesn't allow for years before 2004?

```
a. void Employee::set_hire_year(int hire_year_param) {
    if (hire_year_param >= 2004) {
        hire_year = hire_year_param;
    }
}
b. void Employee::set_hire_year(int hire_year_param) {
    if (hire_year_param < 2004) {
        cout << "Year must be 2004 or later.\n";
    } else {
        hire_year = hire_year_param;
    }
}
c. void Employee::set_hire_year(int hire_year_param) {
    if (hire_year_param < 2004) {
        cout << "Year must be 2004 or later.\n";
    }
    hire_year = hire_year_param;
}
d. void Employee::set_hire_year(int hire_year_param) {
    if (hire_year_param < 2004) {
```

```
        throw invalid_argument("Year must be 2004 or later.");
    }
    hire_year = hire_year_param;
}
```

3. ♢ Which of the following functions would be the best way to overload this setter function so it can accept a string?

```
void Employee::set_salary(double salary_param) {
    if (salary_param <= 0) {
        throw invalid_argument("Salary must be greater than 0.");
    }
    salary = salary_param;
}
```

```
a. void Employee::set_salary(string salary_param) {
    salary = stod(salary_param);
}
b. void Employee::set_salary(string salary_param) {
    double annual_salary = salary_param;
    salary = stod(annual_salary);
}
c. void Employee::set_salary(string salary_param) {
    double annual_salary = stod(salary_param);
    set_salary(annual_salary);
}
d. void Employee::set_salary(string salary_param) {
    double annual_salary = stod(salary_param);
    salary = annual_salary;
}
```

4. ♢ What keyword do you use in a function declaration to indicate that the function can be overridden?
   a. override
   b. virtual
   c. abstract
   d. base

```
Inheritance Hierarchy A
-----------------------------------------------------------------
Vehicle
    MotorVehicle
        Automobile
        Motorcycle
    WaterCraft
        Sailboat
        Canoe
```

5. ♢ (Refer to Inheritance Hierarchy A.) If the Vehicle class contains functions named get_retail_price() and get_description() and the get_description() function is overridden by every subclass in the hierarchy, which get_description() function is executed by the following code?

```
void display_vehicle(const Vehicle& v) {
    cout << "Description:  " << v.get_description() << '\n'
         << "Retail price: " << v.get_retail_price() << "\n\n";
}

int main() {
    Motorcycle motorcycle("Harley-Davidson FXDR 114", 21349.0);
    display_vehicle(motorcycle);
}
```

a. the one in the Vehicle class
b. the one in the MotorVehicle class
c. the one in the Motorcycle class
d. none of the above because the display_vehicle() function only accepts a Vehicle object

6. ♢ (Refer to Inheritance Hierarchy A.) If the MotorVehicle class contains a protected function named get_engine_type(), what other class or classes can access this function?
a. Vehicle, Automobile, and Motorcycle
b. Automobile and Motorcycle
c. Vehicle
d. Vehicle and WaterCraft
e. no other classes can access it

7. ♢ Which of the following is an advantage of using static functions?
a. The functions can easily access the data members of an object.
b. The code that calls the functions is more concise.
c. You don't have to create an object from the class to call the static functions.
d. All of the above
e. a and c only

8. ♢ Which of the following statements is not true about a friend function?
a. It's defined outside the classes that declare it.
b. It can directly access the data members of the classes that declare it as a friend function.
c. It's typically used when public setter and getter functions aren't provided for private data members.
d. It breaks encapsulation.

9. ♠ **Task List**: Create an object-oriented program that allows you to manage a task list that's stored in a text file. Save in folder `lab8-q9`.

Console

```
Task List

COMMANDS
v - View pending tasks
a - Add a task
c - Complete a task
h - History of completed tasks
x - Exit

Command: v
1. Buy toothbrush
2. Do homework

Command: c
Number: 2

Command: a
Description: Pay Bills

Command: v
1. Buy toothbrush
2. Pay Bills

Command: h
1. Get bike fixed (DONE!)
2. Call your mom (DONE!)
3. Do homework (DONE!)

Command: x
Bye!
```

Specifications

- Use a Task class to store the description of the task and whether it has been completed. Include an insertion operator (<<) operator to make it easy to display the task on the console.
- Use a TaskList class to store a list of Task objects. Include a += operator that adds a task to the list and a subscript operator ([ ]) that gets a task from the specified index in the list.
- Use a TaskIO class to store two static functions that work with the text file that stores the data for the program. One function should read data from the file and store it in a TaskList object. The other should write the data in the TaskList object to the file.
- The view command should only display tasks that have not been completed.
- The add command should add a task to the list.

- The complete command should mark a task as completed.
- The history command should display tasks that have been completed.
- You can assume the user will enter a valid integer for the task number, but you should check to make sure that number is within a valid range.