UNM | Computer Engineering

ECE 330 Software Design

Homework 4

The objective of this homework is to give you experience using inheritance with C++. You will use the provided employee base class and implement two derived classes, salaried employee and hourly employee. This assignment will be used for a follow-on assignment.

For this assignment, you will use the base class Employee, which is implemented with the files Employee.h and Employee.cpp. A test file, test.cpp, is also provided for your use, as you choose to use it.

Each employee object has a user id; a first and last name; a middle initial; and a department code (integer). Your implementation must now add two derived classes. The first one will be for a salaried employee, which will require a monthly salary variable. You will need member functions including a constructor, set and get salary functions, a salary calculation function, and a print function. For versatility, you might include a variable that specifies what fraction of time the person worked and use the fraction in the salary calculation. Hint: set the fraction default to one. For consistency, name your salaried employee class as SalariedEmployee

Your second class should represent an hourly worker. In this case you will need to store the hours worked and the hourly rate. You should also include provisions for overtime hours, which will be paid at 1.5 times the regular hourly rate. Hint: set the default overtime hours to 0. You will need similar functions as the salaried employee to set and get variables, as well as to calculate salary. Name your hourly employee class as HourlyEmployee.

Generate a test file that includes at least two of each type of worker to test the classes for proper operation. I recommend a full-time and half-time salaried worker and two hourly worker test cases, one of which earns overtime.

Execute your test program and copy the outputs to a text file to demonstrate proper execution.

What to submit:

Please submit a copy of your source code files and a text file(s) that includes execution output that demonstrates proper operation.