Lab Assignment #4: Introduction to Objects

1. Objectives

The objective of this assignment is to provide you with a practical introduction to programming with C++ objects. This assignment requires you to complete the implementation of two simple classes and their associated member functions, as well as performing basic I/O operations.

2. Problem Statement

Your task is to implement a simple program for performing the weekly payroll calculations for a company with several different stores. Each store has a staff, consisting of one or more employees; each of the employees receives a different hourly wage for their work, and works a different number of hours per week.

You are to implement two classes, as described below. For **this assignment only**, you are free to add new members to the classes, if required, but you must not change or remove any of the class elements that we have provided (*e.g.*, you cannot change any of the provided members from “private” to “public”, change their data types, parameter lists, etc.). Any variables that you add to a class must be “private”; if you need to access the variables outside of your class, you should write “public” functions to read them or change them.

The Employee Class

An *Employee* object exists for each employee. The object contains the employee's name, their hourly pay rate, and the number of hours they worked this week. Its declaration is provided to you in Employee.h, and you should fully define the class in Employee.cpp.

```
class Employee
{
        private:
                char *          name;           // Employee's name
                double          hourlyRate;     // Hourly pay rate
                double          hours;          // Hours worked this week

                double          taxRate ();

        public:
                Employee ();
                ~Employee ();

                void            setName (char * _name);
                void            setHourlyRate (double _hourlyRate);
                void            setHours (double _hours);
                void            printPayroll ();
};
```

The employees' wages are paid and taxed, according to the following rules:

1. Employees are paid "time and a half" (*i.e.*, 1.5 times their normal salary) for any work in excess of 40 hours/week. For example: an employee working 42 hours at $20/hour would receive a total of ((40 hours · $20/hour) + (2 hours · $20/hour · 1.5)) = $860, before taxes.

2. An employee's expected annual income is calculated by assuming that they work 40 hours/week for the entire year. For example: every employee who is paid $17/hour would have an *expected annual income* of ($17/hour · 40 hours/week · 52 weeks/year) = $35,360.

3. Employees are taxed at the following rates, based on their *expected annual income*:

Expected Annual Income	**Tax Rate**
Less than $20,000	15%
20,000 – 29,999.99	20%
30,000 – 39,999.99	25%
40,000 – 49,999.99	30%
50,000 – 59,999.99	35%
60,000 – 69,999.99	40%
$70,000 or above	45%

The `Employee::printPayroll` function should print out the employee's payroll information. (Please see the *Input and Output* section, below, for a description of the output format.)

The Staff Class

Each *Staff* object contains an array of *Employee* objects large enough to hold (MAX_EMPLOYEES) entries. The class declaration is provided to you in Staff.h, and you should fully define the class in Staff.cpp.

```
class Staff
{
        private:
                Employee        employees[MAX_EMPLOYEES];

        public:
                Staff ();
                ~Staff ();

                void            addEmployee (char * _name,
                                       double _hourlyRate, double _hours);
                void            printPayroll ();
};
```

The `Staff::addEmployee` function is used to populate the `employees` array. You may find it necessary to expand the Staff class, in order to keep track of the number of employees you have added.

3. Input and Output

All output and error messages should be printed to standard output (`cout`), and all input should be read from standard input (`cin`). Your program should accept the following commands:

new *storeNumber firstName lastName hourlyRate hoursWorked*

Adds a new employee. Each employee works for a different store: they should be added to the Staff of store number *storeNumber*. (You should keep track of each store's employees in a separate Staff object; the value of *storeNumber* will range from 0 to [`MAX_STORES – 1`].) The employee's name (both *firstName* and *lastName*) should be combined together and used to initialize the `name` field of the Employee object. The *hourlyRate* and *hoursWorked* should be used to initialize the `hourlyRate` and `hours` fields of the Employee object. This command should print "New: OK" after adding the employee.

payroll *storeNumber*

Prints the payroll for the employees that belong to the specified store. See the example, below, for the format of the payroll output. If the payroll is printed successfully, the command should end by printing "Payroll: OK". If we could not print the payroll (because there were no employees) the command should print "Error: no employees."

As in Assignment #3, the input will be terminated with an end of file `(eof)`.

The following is an example of input commands and output messages. Input fields are shown in **bold**, while output and error messages are highlighted in yellow:

```
new 1 Mary Brown 15.00 42.5
New: OK
new 0 John Smith 15.00 36
New: OK
payroll 0
John Smith: 540.00 - 135.00 = 405.00
Payroll: OK
payroll 1
Mary Brown: 656.25 - 164.06 = 492.19
Payroll: OK
payroll 2
Error: no employees.
(eof)
```

As shown above, each line of the “payroll” output should appear in the following format:

employeeName: *grossPay* – *taxAmount* = *netPay*

Where *grossPay* is the employee's pay before tax, and *netPay* is the employee's pay after tax. Please note: there should be **only** one space after the “:”, and **only** one space on either side of the “-” and the “=”. All dollar amounts should print with two decimal places, and without a dollar-sign (*e.g.*, $1 should print as “1.00”).

The purpose of this assignment is to test your understanding of C++ classes. As such, you may write your assignment with the assumption that we will not test your program with undefined or unreasonable input (*i.e.*, invalid commands, missing fields, extra fields, negative values, more than “MAX_STORES” stores, more than “MAX_EMPLOYEES” employees per store, etc.). You may also assume that there will be only one command per line, and that the individual commands will not be split into multiple lines.

4. Procedure

Create a sub-directory called `lab4` in your `ece244` directory, using the `mkdir` command, and protect it with the `chmod` command. Make it your working directory, and then download the Lab 4 source files that are provided on the course website. Using these files as a starting point, generate a solution to this assignment. Make a `main` function in a file named “main.cpp”, and add other functions to your “main.cpp” that will accept and respond appropriately to the input commands. Additionally, create a Makefile that will produce a `lab4` executable when you type “make”.

5. Deliverables

Your lab submission must consist of the following six files:

- **Employee.h -** The declaration of your Employee class.
- **Staff.h -** The declaration of your Staff class.

- **Employee.cpp -** The implementation of your Employee class.
- **Staff.cpp -** The implementation of your Staff class.

- **main.cpp -** Your code to accept commands and call the appropriate Staff functions.

- **Makefile –** A makefile which will produce an executable called “lab4”

Please note that it is **ESSENTIAL** that your Makefile produce an executable called `lab4` when it is run, and that your `lab4` accept input and produce output **EXACTLY** as specified in the assignment description, without adding any additional output. Please submit your files by using the following command:

```
submitece244f 4 Employee.h Employee.cpp Staff.h Staff.cpp main.cpp Makefile
```

Please only submit the requested six files.