Assignment 10 - Payroll Version 3.0

Your job is to update your payroll program for Armadillo Automotive Group to use a C++ class. Each employee class object should hold the master file information for one employee. You can assume that the company has exactly 6 employees. Use an array of employee objects to hold the master file information for the company employees.
Do not put any pay information, including hours worked, in an Employee object. You might want to create a paycheck struct or object to hold pay information for one employee (this could include the hours worked).

DO NOT DO ANY INPUT OR OUTPUT IN ANY CLASS MEMBER FUNCTION.
The employee information and hours worked will come from input files instead of from the keyboard.

Employee class

Create a class to represent the master file information for one employee. Start with the following partial Employee class. You will need to add a "get" function for each class data member to return its value.

```
class Employee
{
  private:
    int id;             // employee ID
    string name;        // employee name
    double hourlyPay;   // pay per hour
    int numDeps;        // number of dependents
    int type;           // employee type

  public:
    Employee( int initId=0, string initName="",
              double initHourlyPay=0.0,
              int initNumDeps=0, int initType=0 );  // Constructor

    bool set(int newId, string newName, double newHourlyPay,
             int newNumDeps, int newType);

};

Employee::Employee( int initId, string initName,
                    double initHourlyPay,
                    int initNumDeps, int initType )
{
  bool status = set( initId, initName, initHourlyPay,
                     initNumDeps, initType );

  if ( !status )
  {
    id = 0;
    name = "";
    hourlyPay = 0.0;
    numDeps = 0;
    type = 0;
  }
}

bool Employee::set( int newId, string newName, double newHourlyPay,
                              int newNumDeps, int newType )
{
  bool status = false;

  if ( newId > 0 && newHourlyPay > 0 && newNumDeps >= 0 &&
       newType >= 0 && newType <= 1 )
  {
    status = true;
    id = newId;
    name = newName;
    hourlyPay = newHourlyPay;
    numDeps = newNumDeps;
    type = newType;
  }
  return status;
}
```

- Note that the constructor and set functions do validation on the data that is to be stored in the Employee object.
- You should be able to copy this class into your editor by highlighting the code, making a copy of it (ctrl-c in Windows), and then pasting the code into your editor window.
- Do not make any changes to the data members of the class. Do not add any new data members to the class. Do not make any changes to the constructor and set functions.
- To complete the class, add a "get" function for each of the private data members (that is, 5 functions). Each get function should return the value of a data member.
-

<u>Program input</u>
The program input consists of two files - a master file and a transaction file. Your code must work for the 2 input files provided. You may also want to test your program with other input data.

<u>Master file</u>
The master file has one line of input per employee containing:
- employee ID number (integer value)
- name (20 characters) - see Hint 6 below on how to input the name

- pay rate per hour (floating-point value)
- number of dependents (integer value)
- type of employee (0 for union, 1 for management)
-

This file is ordered by ID number and contains information for 6 employees. You can assume that there is exactly one space between the employee ID number and the name. You can also assume that the name occupies 20 columns in the file. **Important:** See the **Requirements/Hints** section at the bottom of this page for more information on the input files.

Transaction file (weekly timesheet information)

The transaction file has one line for each employee containing:

- number of hours worked for the week (floating-point value)

This file is also ordered by employee ID number and contains information for the 6 employees. Note: You can assume that the master file and the transaction file have the same number of records, and that the first hours worked is for the first employee, etc. You can also assume that the employee IDs in the master file are exactly the same as the employee IDs in the transaction file. **Important:** See the **Requirements/Hints** section at the bottom of this page for more information on the input files.

Calculations

- Gross Pay - Union members are paid 1.5 times their normal pay rate for any hours worked over 40. Management employees are paid their normal pay rate for all hours worked (they are not paid extra for hours over 40).
- Tax - All employees pay a flat 15% income tax.
- Insurance - The company pays for insurance for the employee. Employees are required to buy insurance for their dependents at a price of $30 per dependent.
- Net Pay is Gross Pay minus Tax minus Insurance.

Payroll Processing

Notice that when you store employee master information in an Employee object, the set() function does data validation. If any of the employee master information is invalid, the set() function stores default values in the Employee object. In particular, the ID of the employee is set to zero.

When processing the payroll:

- If the employee master information for the employee is invalid (if the ID is 0), print an appropriate error message on the screen and do not pay the employee. The employee should not appear in the Payroll Report.
- If the hours worked for an employee is invalid (less than 0.0), print an appropriate error message on the screen. The employee should not be paid and should not appear in the Payroll Report.
- When all employees have been processed, print on the screen the total number of transactions that were processed correctly during the run.

Payroll Report

This report should be printed to a file. It should not be printed on the screen. The payroll report should be printed in a tabular (row and column) format with each column clearly labeled. Do not use tabs to align your columns - you need to use the setw() manipulator. Print one line for each transaction that contains:

- employee ID number
- name
- gross pay
- tax
- insurance
- net pay

The final line of the payroll report should print the total gross pay for all employees, and the total net pay for all employees.

Requirements/Hints:

1. Global variables are variables that are declared outside any function. **Do not use global variables in your programs.** Declare all your variables inside functions
2. A sample Master file and Transaction file are included.
   Your program must work correctly for these files.
   Maybe the best way to copy a file to your computer is to right-click on the link, then choose "Save As" or "Save Link As" from the popup menu. Optionally you may be able to open the text file in your browser and select "Save As" or "Save Page As" from your browser menu.
   If you create your own test files in a text editor, be sure to press the enter key after the last line of input and before you save your text. If you choose to copy the text from my sample file and paste it into a text editor, be sure to press the enter key after the last line of input and before you same your text.
3. Use the C++ **string** class to hold the employee name.
4. You should use an Employee class object to hold the master file information for one employee.
5. The Payroll Report should be written to a file.
6. Notes on reading C++ string objects:

   The getline function is first introduced in Chapter 3 and then covered more thoroughly in the Files chapter. The C++ code:

   ```
   string name;
   getline( cin, name );
   ```

   will read all characters up to the end of the line (to the first newline character in the input stream). You can specify a character other than the newline character to stop the input. For example:

   ```
   getline( cin, name, '#' );
   ```

   will read all characters until the '#' is found in the input. The Transaction file uses the '#' to mark the end of the names so that they are all 20 characters long.

Note that you can use getline with input file streams by replacing cin with the input file object.

Master9.txt

```
5 Christine Kim       # 30.00 3 1
15 Ray Allrich        # 10.25 0 0
16 Adrian Bailey      # 12.50 0 0
17 Juan Gonzales      # 30.00 1 1
18 J. P. Morgan       #  8.95 0 0
22 Cindy Burke        # 15.00 1 0
```

Trans9.txt

```
5 40.0
15 42.0
16 40.0
17 41.5
18 -40.0
22 20.0
```