Assignment 9 - Payroll Version 2.0

- Your program should handle any number of employees up to 100. The employee master data will be input from a file (1 to 100 employees).
- Employee timecard information (employee id and hours worked) will be in another input file.
- Use a C-string (character array) instead of a C++ string object to store an employee name.

Important Note: When you are reading input from a file using an end-of-file loop, it is critical that the last line of the file be terminated with a newline character. The description of the end-of-file loop and all the example code in the textbook assume that the last line of input is terminated with a newline. The examples may not work correctly for files that are missing a newline at the end.

By convention all files should be terminated with a newline character.

Employee class

You will use your employee class. Change the member variable for the employee name so that you use a C-string instead of a C++ string object. Your C-string should be able to hold names up to 25 characters in length. Then modify your member functions as required to work with the C-string.

```
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;
}
```

Program input

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.

Master file

The master file has one line of input per employee containing:

- employee ID number
- name of up to 20 characters
- pay rate per hour
- number of dependents
- type of employee (0 for union, 1 for management)
- gender of employee (M or F)
-

Notes:

This file contains one extra piece of data at the end of each line - a single character ('M' or 'F'). You can ignore this data.

Validation for the data in this file is done in the set member function of the Employee class.

This file is ordered by ID number. Your code should work for any number of employees up to 100 (use an end-of-file loop to read this file). 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.

Transaction file (weekly timesheet information)

The transaction file has one line for each employee who worked containing:

- employee id number (an integer greater than 0)
- number of hours worked for the week (a floating-point value of 0.0 or greater)

This file may have any number of transactions and may be in any order.

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

Since the employee master information and the timecard information (hours worked) are not input in the same order, and the total number of employees likely will be different than the number of timesheets, trying to use parallel arrays for this assignment will get very tricky. A better strategy might be to:

- read the employee master information into an array (count the number of employees as you read)
- for each timecard record in the transaction file, try to look up the matching employee in the employee array and process as required.

Program output

The program output will consist of two reports:

Error and Control Report

This report can be printed on the screen or alternatively can be printed to a file.

- Prints the transaction file input lines for which there is no matching master file record or the hours worked is invalid (a negative number). These lines should be ignored (no further processing is done for the line) and processing continues with the next transaction.
- Prints 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. Print one line for each transaction that contains:

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

All dollar amounts should be displayed with 2 decimal places. The decimal points should line up vertically in your columns (use the setw() manipulator when printing columns).
The final line of the payroll report should print the total amount of gross pay and total amount of net pay for the week.

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. You can assume that the Master file does not contain errors (error checking is done by another program). Error checking for the Transaction file is described in the Error and Control Report.

   Your program must work correctly for these files.

   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 C-strings (not the C++ **string** class) to represent strings in your program.
4. You should use a class to represent the master file information for one employee.
5. The Payroll Report must be written to a file.
6. The **master file** information should be read into an array.
7. You do not need to store the **transaction** information in an array. You can read each transaction (timecard information), look up the matching employee id in the employee master array, and then calculate and print the paycheck information.
8. Your program might be structured similar to the attached sample program:
9. Notes on reading c-strings:
   The cin.getline function is first introduced in chapter 3 and then covered more thoroughly in the Files chapter. The C++ code:

   ```
   char name[21];
   cin.getline( name, 21, '\n' );
   ```

   should read the next 20 characters in the input stream, or up to the first newline character, whichever comes first, and store the characters in name followed by the null terminator character '\0'. This code works fine if you are reading a string at the end of a line.

   ```
   char name[21];
   cin.get( name, 21, '\n' );
   ```

Master10.txt

```
5 Christine Kim       30.00 2 1 F
15 Ray Allrich        10.25 0 0 M
16 Adrian Bailey      12.50 0 0 F
17 Juan Gonzales      30.00 1 1 M
18 Morris Kramer       8.95 0 0 M
22 Cindy Burke        15.00 1 0 F
24 Esther Bianco      10.25 0 0 F
25 Jim Moore          27.50 3 1 M
32 Paula Cameron      14.50 0 0 F
36 Melvin Ducan       10.25 0 0 M
37 Nina Kamran        30.00 1 1 F
38 Julie Brown        35.00 0 1 F
40 Imelda Buentello   14.50 0 0 F
42 J. P. Morgan       12.50 0 0 M
43 Maria Diaz         15.00 0 0 F
```

Trans10.txt

17 20.0
37 31.0
5 40.0
42 39.5
24 15.0
22 40.0
15 42.0
18 40.0
25 45.0
32 25.0
40 47.5
36 -40.0
16 40.0
38 35.0
43 40.0
33 30.0