Project -06 - Payroll Version 1.0

In this assignment you must create and use a struct to hold the general employee information for one employee. Ideally, you should use an array of structs to hold the employee information for all employees. If you choose to declare a separate struct for each employee, I will not deduct any points. However, I strongly recommend that you use an array of structs.

Your job is to write a payroll program for Armadillo Automotive Group.

Warning: Input using the extractor operator (">>") is simple. However, when you mix the extractor operator with the getline function, things get tricky. Be sure that you understand how input works and review the examples of using the getline function in your textbook.

Program input

For a payroll program you would normally input the data from files, but in this first version you will input the data from the keyboard. The program input should be in 2 parts: employee master information and timesheet information.

Employee Master information

The employee master information consists of the following data:

- employee ID number (integer value)
- employee name (your program should handle names of up to 20 characters - may contain spaces)
- pay rate per hour (floating-point value)
- type of employee (0 for union, 1 for management)

Note: Use the C++ **string** class for the employee name. Use a C++ **struct** to hold the employee master information for one employee. **Do not put Timesheet information in the employee master information struct.**

Timesheet information

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

Assume that there are exactly 4 employees. Your program should first input the employee master information into an array of structs. Then use a separate loop to do the payroll processing for each employee (input the employee's hours worked and calculate their pay (see the example program dialog below).

Calculations

- Gross Pay - Union members are paid their normal pay rate for the first 40 hours worked, and 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 paid for overtime hours, but they are paid at their normal hourly rate).
- Tax - All employees pay a flat 15% income tax.
- Net Pay is Gross Pay - Tax.

Input validation

The input should be checked for reasonable values. If a value is not reasonable, your program should print an informative error message and ask the user to re-enter the value.

- The following data should be positive numbers (greater than 0): employee id and pay rate.
- The following data should be non-negative (0 or larger): hours worked.
- Employee type should be 0 or 1.

Program output - Payroll Report

Your program should gather all the required input before the payroll report is printed. The input prompts must not be mixed in with the report. If necessary, you can print the payroll report to an output file.

The payroll report should be formatted in a tabular (row and column) format with each column clearly labeled with a column heading. All dollar amounts should be formatted with 2 decimal places. Note: do not use tabs between the columns - use the setw manipulator to set the column width so that you can line up columns of numbers on the decimal point. Print one line for each transaction that contains:

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

The final lines of the payroll report should print the total amount of gross pay and total amount of net pay for the week (the total for all employees).

Other Requirements

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. Use the C++ **string** class to represent strings in your program.

3. You should use a struct to represent the employee master information for one employee. **Note:** you should NOT include Timecard information in this struct. Timecard information may change from one pay period to the next while employee master information usually does not. In other words, an employee's master information and information about a specific paycheck are 2 different things and logically should not be combined in the same struct (or object).
4. The timecard information (hours worked) does not need to be stored in an array.

Example

SAMPLE RUN (User entry is in RED)

```
Enter information for employee 1
Employee id: 22
Employee name: Cindy Burke
Pay rate: 15.00
Type: 0

Enter information for employee 2
Employee id: 42
Employee name: J. P. Morgan
Pay rate: 12.50
Type: 0
...
(input employee master information for last 2 employees)
...
Enter timecard information for each employee:
Hours worked for Cindy Burke: 40.0
Hours worked for J. P. Morgan: 39.5
...
(input timecard information for last 2 employees)

...

Payroll Report

ID  Name                        Gross Pay      Tax  Net Pay
22  Cindy Burke                    600.00    90.00   510.00
42  J. P. Morgan                   487.50    73.13   414.38
41  Sue Kim                        665.00    99.75   565.25
45  Ernest Chavez                  760.00   114.00   646.00

Total Gross Pay $ 2512.50
Total Net Pay   $ 2135.63

Press any key to continue . . .
```