COP3223 Introduction to C - Program 4
Payroll – W2 Forms

Objectives
1. To give students practice in using structures, and arrays of structures.

The Problem: Payroll – W2 Forms
Companies have to issue W2 forms for all of their employees for tax purposes. In this program, you will write a program that assists a company in creating W2 forms.

Many companies have a system where employees clock in and clock out to determine how much time they work each week. Since these records are already automatically stored in files on companies' computers, it only makes sense to use these files to help calculate how much each employee should get paid. (Once this is calculated, then tax information for the W2 form can also be calculated.) In order to determine how much an employee gets paid each week, you need to know the following two pieces of information:

1) The number of hours they worked in a week
2) Their hourly pay rate

If an employee works 40 or less hours a week, then s/he simply gets paid the number of hours s/he worked times the hourly pay rate. If an employee works more than 40 hours, then s/he gets paid an extra 50% of the regular pay for the hours over 40 s/he worked. (For example, if an employee's pay rate is $10/hour and the employee works 50 hours, then s/he would get paid $550, because s/he gets $10x40 = 400 normally, plus an extra 1.5x(10 hr)x($10/hr) = $150 for his/her last ten hours of work.

Following this calculation, you will also have to calculate the amount of money withheld for taxes. Here is the rule that will be used to calculate how much money is withheld for taxes each week:

1) 10% of regular pay (pay for the first 40 hours) will be withheld for taxes.
2) 20% of overtime pay (pay for any time beyond 40 hours) will be withheld for taxes.

For example, if an employee's pay rate is $10/hour and the employee works 50 hours, then s/he would get $70 withheld for taxes because s/he got $400 (40hrs x $10/hr) regular pay from which $40 is withheld, and s/he got $150 (10hrs x $15/hr) overtime pay from which $30 is withheld.

You are to write a program that reads in the raw time data from the file "clock.txt", processes the W2 forms and outputs the final forms to the file "w2.txt". The input file will consist of all employee data for several weeks. Your output file contains W2 "forms" for each employee. You are guaranteed that the file will contain data for no more than 20 employees.

<u>Format of clock.txt</u>
The first line of the file will contain a single positive integer, *n*, in between 1 and 20, inclusive, which represents the number of employees at the company. The next *n* lines of the file will contain the first name of the current employee, the last name of the current employee and a single positive real number representing his/her hourly pay rate. Each name will contain fewer than 30 characters. Each piece of data on the line will be separated by a space. *(Note: No two employees will have the same exact first AND last names. Thus, there can't be two Steve Smith's but there can be a Steve Jones and a Steve Smith.)* The next line of the file will contain a single positive integer, *k*, which represents the total number of sets of data contained in the file. Each set of data represents the data for a single week. The first line of each set of data will contain a single positive integer *m*, representing the number of employee records for the week. (Note: a single employee record denotes one instance of an employee clocking in and clocking out.) The following *m* lines will contain one employee record each. Each employee record will have the following format:

```
LastName FirstName HrIn MinIn HrOut MinOut
```

The first two pieces of information will be strings storing the last name and first name, respectively, of the employee clocking in and out. (*Note: You are guaranteed that these names will match EXACTLY one employee initially listed in the file.*) The last four values on a line will be integers. The first integer, `HrIn`, represents the hours (in military time) that the given employee clocked in to work. This value is guaranteed to be in between 0 and 23, inclusive. The second integer, `MinIn`, represents the minutes (in military time) that the given employee clocked in to work. This value is guaranteed to be in between 0 and 59, inclusive. The last two values, `HrOut` and `MinOut`, represent the hours and minutes respectively (in military time), that the given employee clocked out of work. You are guaranteed that this second time occurs later in the day than the first. (Thus, no employee EVER works through midnight.) Thus, the following line:

```
Williams Steve 8 0 16 30
```

means that employee Steve Williams worked from 8am to 4:30pm, for a total of 8.5 hours.

<u>Format of w2.txt</u>
The first line of the output file will contain the following

```
Number of employees: n
```

where *n* is the number of employees in the company. The second line will be blank. Following that will be *n* sets of data, one for each employee in the input file. Each set of data will be separated by two blank lines. The first two lines of each set of data will read:

```
W2 Form
-------
```

The third line of each set of data will have the following format:

```
Name: First Last
```

where First and Last are the first and last names of the current employee, respectively. (The employees should be printed out in the order in which they were originally read in from the input file.)

The fourth line for each set of data should have the following format:

```
Gross Pay: XXX.XX
```

where XXX.XX represents the gross pay (in dollars) for the current employee rounded to the nearest cent.

The fifth line for each set of data should have the following format:

```
Taxes Withheld: XXX.XX
```

where XXX.XX represents the taxes withheld (in dollars) for the current employee rounded to the nearest cent over the whole pay period specified in the input file.

The last line of each data set will be of the format:

```
Net Pay: XXX.XX
```

where XXX.XX represents the net pay for the current employee rounded to the nearest cent over the whole pay period specified in the input file. (The net pay is simply the gross pay minus the taxes withheld.)

Implementation Requirements

You must utilize the following struct to store information about a single employee:

```
struct employee {
  char first[MAX_LEN];
  char last[MAX_LEN];
  double payperhr;
  double gross;
  double taxes;
  double hours_in_week;
}
```

In particular, the first three fields will store the standard information about the employee. The next two fields will be updated at the end of each work week and will store the adjusted gross pay and taxes withheld. Finally, the last component keeps track of the number of hours the employee has worked in the current week. (Thus, this needs to get reset to 0 after finishing reading in each week's information. The reason for this is that the amount of tax withheld can ONLY be determined after reading through a whole week's worth of information – thus, hours_in_week will be updated numerous times while reading in information for a single week. But, gross and taxes will only be updated after a whole week's worth of data has been read in. Also, hours_in_week has to be reset to 0 at the end of each week.)

Furthermore, your main must contain an array of type struct employee of size 20. (Note: 20 should be stored in a constant that is declared with a #define. Also, define MAX_LEN as a constant equal to 30 and be used for the strings, since each name is guaranteed to be fewer than 30 characters long.)

Example Input and Output Files
These will be posted on the section's webpage as individual text files.

References
Lecture Notes: Structures and examples using arrays of structures, Files, and Functions.

Grading Notes
In order to receive full credit, not only does your program have to work properly, but you must do the following:

1) Use multiple functions (at least one function other than main)
2) Define constants for appropriate values (you need to figure out what these are)

Restrictions
Name the file you create and turn in *assign4tax.c*. Although you may use other compilers, your program must compile and run using Codeblocks or a compatible compiler. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments in your code, when appropriate. If you have any questions about this, please see a TA.

Input Specification
Assume that the input file that we use for testing adheres to all the specifications described above. (*In essence, you do not have to error check the input file at all.*)

Output Specification
Your output file should follow the format described above.

Deliverables
A single source file named *assign4tax.c* turned in through WebCourses.