Homework 5

COP 3223C Introduction to Programming with C, Section 0V06

Spring 2021

1 Description

For this assignment you will be working with C structures and files. You will write two functions named `PrintDeptPayrollRecords(…)`, and SaveDeptPayrollRecordsToFile(…). **This time your `main()` function will also be graded.** All three functions should be in a single C source file named **homework5.c**. The descriptions of each function are listed below in the Function Requirements Section. Feel free to implement additional "helper" functions as you see fit.

2 Deliverables

A single C source file named homework1.c must be submitted to Webcourses by the assignment deadline (posted in Webcourses).

3 Grading Rubric

Uses correct filename (See Deliverables)	5 pts
Includes appropriate header comment (See Style Requirements)	5 pts
Uses tasteful comments (See Style Requirements)	10 pts
Follows all style requirements (See Style Requirements)	20 pts
Required functions work correctly and produce correct output (See Function Requirements)	60 pts
Total	**100 pts**

4 SUPER IMPORTANT INFORMATION

- **Sharing code or posting assignment details in public places will be considered academic dishonesty and will result in an automatic 0 grade for this assignment.** Feel free to have *high level* discussions about the assignment and your solution with your classmates. In general, discussions about the assignment are encouraged if they are only with students actively enrolled in this course and do not include any sharing of code.
- **Copying source code from the internet or other sources other than your own brain will be considered academic dishonesty and will result in an automatic 0 grade for this assignment.**
- Your source file must be named correctly to receive full credit. See Deliverables for the required filename. If you submit your file multiple times to Webcourses an additional "-n" will be added to the end of the filename (example: homework5.c, homework5-1.c, homework5-2.c, etc.). Files with this suffix will also be accepted.
- Your source file must contain the exact method signatures listed in Function Requirements to receive full credit.
- Your source file must compile and run to receive credit. **Submissions that do not compile will receive an automatic 0 grade.**
- **Submissions that print extra information to the console will not receive full credit.** Your submission should only produce console output if specifically requested in Function Requirements.

5 Function Requirements

The source file you submit should contain the following functions. Both of them work with the structure described below.

You need to store and manage Payroll Records stored as an array of structures of the following form:

```
struct payroll {
	unsigned long long int internal_id;
	char first_name[50];
	char last_name[50];
	char department[100];
	unsigned char dob_day;
	unsigned char dob_month;
	unsigned short dob_year;
	double monthly_salary;
};
```

The array of structures is stored as an ***external variable***

```
#define MAX_PAY_RECS_NO 10000
struct payroll pay_recs[MAX_PAY_RECS_NO];
```

The number of ***actual*** payroll records is stored in the following ***external variable***:

```
size_t pay_recs_no;
```

Your responsibility is to develop a function that prints out the payroll records of employees of a certain department, the number of employees in the department and their aggregated monthly salary (sum of monthly salaries of the department's employees). The output should be made in the following format:

```
Employees of the department_name Department:
Last Name       First Name      Date of Birth   Monthly Salary
------------------------------------------------------------------------------------------------------------
Smith           John            Jan 20, 1997    5920.00
Gary            Richard         Jul 3, 1991     6530.00
Peterson        Jason           March 5, 1995   6200.00
Total Number of Employees in the Department: 3
Aggregated Salary of the Department: 18650.00
```

If the department supplied as a parameter of the function has no payroll records, the function shall print the following:

```
Department department_name has no payroll records.
```

This is the function's signature:

```
void PrintDeptPayrollRecords(char* dept_name);
```

The second function's signature is:

```
void SaveDeptPayrollRecordsToFile(char* dept_name, char* file_name);
```

This function does the same as `PrintDeptPayrollRecords(…)` does, but instead of printing out the results on screen, it saves them to the file with file name corresponding to the `file_name` parameter.

In the `main()` function, please do the following:

1. Initialize 10 or more payroll records in the payroll records array `pay_recs[]`. **Please do NOT forget to set the `pay_recs_no` external variable to the correct value (the actual number of the payroll records in the system);**
2. Call the `PrintDeptPayrollRecords(…)` function with the name of an existing department, and then with non-existing one, and check that a correct output is generated;
3. Ask the user to enter the file name of the file for storing the payroll records in.
4. Call the `SaveDeptPayrollRecordsToFile(…)` function with the name of an existing department, and the file name obtained from the user.

Please make sure that the text file is created successfully, that is open it by any text editor (e.g., Notepad), and check its content.

6 STYLE REQUIREMENTS

- **Header Comments:** Submissions must include a header comment of the following form on the very first line of the file:

```
// <Your Name>
// NID: <Your NID>
// <Assignment Name> “<Filename>”
// <Course ID> <Section #>, < Current Semester>
```

```
// John Smith
// NID: jo123456
// Homework 5 “homework5.c”
// COP 3223C Section 0V06, Spring 2021
```

- **Function Comments:** All required functions must have a comment directly above them that describes what they do. Function comments should include a description of a function’s expected inputs and expected outputs:

```
// Takes a non-negative integer n as input and returns n! where
// n! is a positive integer and  n! = 1 * 2 * 3 * 4 * … * n
int factorial(int n)
{
        …
}
```

- **Tasteful Comments:** You should add comments throughout your code that make the code easier to read. Be careful when adding tasteful comments to your code; only lines that *need* tasteful comments should have them. Tasteful comments should be placed directly above the line or block of code they describe, or on the same line as the line they describe:

```
void foo(int n)
{
        int i, a = 0, b = 1;

        // You can put comments like this :)
        for (i = 0; i < 10; i++)
        {
                a++; // You can put comments like this :)
                b--;

// You can NOT put comments like this (bad indentation)
                a = a * b;
        }
}
```

  Please also **avoid very long comments**. Comments must usually fit one visible line in the editor.

- **Whitespace:** Variables, values, and operators should be separated by a single space in your code:

```
void foo(int n)
{
        int a = 0, b = 1;

        a = a + 1; // This is allowed
        b=b+1; // This is NOT a good style because there are no spaces between b, =, +, and 1
}
```

Whitespace: Your submission should include blank lines as needed to make your code easier to read. In general, you should include blank lines to separate statements that preform different tasks:

```
0 |void foo(int n)
1 |{
2 |       int i, a = 0, b = 1;                    // Initialize some variables
3 |                                               // Blank line for readability
4 |       for (i = 0; i < 10; i++)                // For loop performing a new task
5 |       {
6 |               a++;
7 |               b--;
8 |       }
9 |}
```

- **Indentation:** Submissions must have consistent indentation. This means that any two lines of code inside the same code block (functions, if statements, loops, etc.) must have the exact same indentation:

```
0 |void foo(int n)
1 |{
2 |       int i, a = 0, b = 1; // Lines 2 and 4 must have the same indentation
3 |
4 |       for (i = 0; i < 10; i++)
5 |       {
6 |               a++; // Both of these lines are in the same for loop,
7 |               b--;  // so they have the same indentation.
8 |       }
9 |}
```

- **Indentation (Contd.):** When a nested code block is created (like a loop inside a function), all lines of code inside the nested code block should be indented one more time than the nested block's container (I know that sounds confusing, so take a look at the example below):

```
0 |void foo(int n) // This line has an indentation of 0 (no indentation)
1 |{
2 |       int i, a = 0, b = 1; // This line has an indentation of 1 because it is inside of foo
3 |
4 |       for (i = 0; i < 10; i++) // This line has an indentation of 1 because it is inside of foo
5 |       {
6 |               a++; // Both of these lines have an indentation of 2
7 |               b--;  // because they are inside a for loop inside of foo.
8 |       }
9 |}
```