Structures

Workshop 5 (out of 10 marks - 6% of your final grade)

In this workshop, you will code a C-language program with an object of structure type.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities:

- to store data of different types using a structure type
- to declare an object of structure type
- to access the members of an object of structure type
- to describe to your instructor what you have learned in completing this workshop

SUBMISSION POLICY

Part 1 is to be completed by 23:59 on Thursday of your workshop week. Part 2 of the workshop is **due no later than the Sunday following the Part 1 assigned date (even if that day is a holiday) by 23:59.**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to back up your work regularly.

Late submission penalties:

- Part 1 (In-lab) portion submitted late, with part 2 (at-home) portion: 0 for part 1. Maximum of 90/90 for part 2 and reflection

- If any of part 1, part 2 or reflection portions is missing the mark will be zero.

PART 1: (10%)

Download or clone workshop 5 (**WS05**) from **https://github.com/Seneca-144100/IPC-Workshops**

Write your code for this segment in the **emp_inlab.c** file provided with the Visual Studio template project inside the **in-lab** folder.

In this workshop segment, you will implement, Add and Display employee data functionality using C structs and arrays.

OVERVIEW

The **emp_inlab.c template file** has the following already implemented instructions:

- Display a menu list as shown in the following inside a do-while loop construct:
    1. Display Employee Information
    2. Add Employee
    0. Exit
- Capture user input for the above options. Store to an `int` variable named "option"
- Ability to iterate multiple menu choices (until 0 is entered) with required selection construct to direct process flow to the required logic/functionality (using switch). Refer to the comments for each case to identify the functionality required.
- Display an error message for invalid menu option selections in the `default` case
- Initial output information for menu options 1 and 2 is provided with the relevant formatting.

You are required to complete the following.

- Define the number of employees **SIZE** to be 2 using the #`define` directive and inserting it between the library directive and the main function definition (you will increase this value later)
- Declare a struct to represent employee data, which has the following information
    - An identification number stored in an `int`
    - Age stored in an `int`
    - Salary stored in a `double`
- Declare an array of Employee `struct`'s named **emp** that can hold **SIZE** elements. Initialize all of the elements and their member variables to be 0. (Hint: You can assign **{ {0} }** to **emp** at the time of declaration).

IMPLEMENT EXIT PROGRAM FUNCTIONALITY IN CASE 0

Print the exiting message.

```
> Exiting Employee Data Program. Good Bye!!! <
```

IMPLEMENT DISPLAY FUNCTIONALITY IN CASE 1

- Display the elements of the **`struct`** **emp** array. Only display the **valid** employee data ("*Employee ID, Employee Age* and *Employee Salary*"). Do not display any other data. **Hint:** If the identification number is positive, then it is **valid** employee data. Use the following formatting in a **`printf`** statement:

  `%6d%9d%11.2lf` ←→ (*Employee ID, Employee Age* and *Employee Salary*)

- After completing the display, enter a newline using a **`printf`** statement.

IMPLEMENT ADD EMPLOYEE FUNCTIONALITY IN CASE 2

- Keep track of the number of valid employees using an **`int`** variable.
- Check if the **`struct`** **emp** array is full.
- If the array is full, display the following error message.

  ```
  > ERROR!!! Maximum Number of Employees Reached <
  ```

- If the array is not full, accept new employee data ("*Employee ID, Employee Age* and *Employee Salary*") as per the program output listed below and store that data in an empty element of the **`struct`** **emp** array. Increment the number of valid employees as required.

PROGRAM COMPLETION

Your program is complete if your output matches the following output. Red numbers show the user's input.

```
---=== EMPLOYEE DATA ===---

1. Display Employee Information
2. Add Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 111
Enter Employee Age: 34
Enter Employee Salary: 78980.88

1. Display Employee Information
```

```
2. Add Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 112
Enter Employee Age: 41
Enter Employee Salary: 65000

1. Display Employee Information
2. Add Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
ERROR!!! Maximum Number of Employees Reached

1. Display Employee Information
2. Add Employee
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========
   111       34   78980.88
   112       41   65000.00

1. Display Employee Information
2. Add Employee
0. Exit

Please select from the above options: 0

Exiting Employee Data Program. Good Bye!!!
```

PART 1 SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above or any information needed.

If not on matrix already, upload your **emp_inlab.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account and follow the instructions (replace profname.proflastname with your professors Seneca userid and replace **NAA** with your section):

~profname.proflastname/submit 144w5/NAA_lab <ENTER>

Please Note

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

PART 2: (40%)

Copy all of the in-lab code into the "**emp_athome.c**" file, which you will find in the **at-home** folder. Implement the following cases to **emp_athome.c** file.

- Change the value of **SIZE** to 4.
- Expand the menu list (printing) to include options 3 & 4 after option 2 with the following
    - Update Employee Salary
    - Remove Employee
- Create two switch-cases for option 3 & 4 after case 2. Do not forget to include `break;` statements at the end of each case.

IMPLEMENT UPDATE EMPLOYEE DATA FUNCTIONALITY IN CASE 3

- Display the following initially

```
> Update Employee Salary <
> ====================== <
```

- Prompt the user for the employee identification number using a `do-while` loop. While the number is not found in the employee array, keep prompting the user.
- Once the number is found, display the current salary for the employee with that identification number and prompt the user to input the new salary. Replace the old salary with the input value.

IMPLEMENT REMOVE EMPLOYEE FUNCTIONALITY IN CASE 4

- Display the following initially

```
> Remove Employee <
> =============== <
```

- Copy and paste from case 3 the `do-while` loop that prompts the user for the employee identification number.
- Once the number is found, display the following message and ONLY change the employee identification number to indicate an empty slot (or invalid record).

```
> Employee [REPLACE WITH EMP ID] will be removed <
```

- Decrement the valid employee count by one
- **DO NOT REORDER** the items in the array!

PROGRAM COMPLETION

Your program is complete, if your output matches the following output. Red numbers show the user's input.

```
---=== EMPLOYEE DATA ===---

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 5

ERROR: Incorrect Option: Try Again

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 222
Enter Employee Age: 22
Enter Employee Salary: 22222.22

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2
```

```
Adding Employee
===============
Enter Employee ID: 333
Enter Employee Age: 33
Enter Employee Salary: 33333.33

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 444
Enter Employee Age: 44
Enter Employee Salary: 44444.44

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 555
Enter Employee Age: 55
Enter Employee Salary: 55555.55

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
ERROR!!! Maximum Number of Employees Reached

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========
   222       22   22222.22
```

```
   333       33   33333.33
   444       44   44444.44
   555       55   55555.55

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 3

Update Employee Salary
======================
Enter Employee ID: 123
*** ERROR: Employee ID not found! ***
Enter Employee ID: 321
*** ERROR: Employee ID not found! ***
Enter Employee ID: 333
The current salary is 33333.33
Enter Employee New Salary: 99999.99

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========
   222       22   22222.22
   333       33   99999.99
   444       44   44444.44
   555       55   55555.55

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 4

Remove Employee
===============
Enter Employee ID: 789
*** ERROR: Employee ID not found! ***
Enter Employee ID: 987
*** ERROR: Employee ID not found! ***
Enter Employee ID: 333
Employee 333 will be removed

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
```

```
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========
   222       22   22222.22
   444       44   44444.44
   555       55   55555.55

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 2

Adding Employee
===============
Enter Employee ID: 666
Enter Employee Age: 66
Enter Employee Salary: 66666.66

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 1

EMP ID  EMP AGE EMP SALARY
======  ======= ==========
   222       22   22222.22
   666       66   66666.66
   444       44   44444.44
   555       55   55555.55

1. Display Employee Information
2. Add Employee
3. Update Employee Salary
4. Remove Employee
0. Exit

Please select from the above options: 0

Exiting Employee Data Program. Good Bye!!!
```

PART 2 REFLECTION (50%)

Please provide answers to the following in a text file named **reflect.txt.**

In three or more paragraphs and a **minimum of 150 words**, explain what you learned while doing this workshop. In addition to what you have learned, **your answer should at least include the following**:

- Using examples from this workshop to justify your point of view, discuss the advantages of using an array of structs versus parallel arrays when dealing with related data.
- In this workshop, the Employee struct is declared in the emp_athome.c file. Where else could it have been declared and what advantages might it have [**Hint**: this is explained in the notes]?

Reflections will be graded based on the published rubric (https://github.com/Seneca-144100/IPC-Workshops/tree/master/WS05/Reflection Rubric.pdf**).**

Example:
An example reflection answer for Workshop #2 is available demonstrating the minimum criteria:
https://github.com/Seneca-144100/IPC-Workshops/tree/master/WS05/Example Reflection-WS_2.pdf

PART 2 SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **emp_athome.c** **and** **reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account and follow the instructions (replace profname.proflastname with your professors Seneca userid and replace **NAA** with your section):

```
~profname.proflastname/submit 144w5/NAA_home <ENTER>
```

Please Note

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.