YORK UNIVERSITY
LASSONDE SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

EECS 2032Z - Introduction to Embedded Systems

Winter 2025

Bonus Assignment
Due Date: Apr 10, 2025

Note:

- Review the course slides of Week 7 and 8. You may need to refer the course slides of Week 5 and 6, if needed.
- The red text is what you type and the blue text is the computer response.
- Save the `.c` file in a folder named **Bonus Assignment**, zip the folder, and submit it on eClass in the Week 9 Assignment, named 'Bonus Assignment'.

Problem

Write a C program that manages employee performance records for a company. The program should store employee information, calculate a performance score, and sort employees based on their performance score in descending order. Additionally, the program should allow employees to be searched by name and updated by `Employee ID`. Each employee has the following attributes:

1. `Name:` A string (up to 100 characters) representing the employee's full name
2. `Employee ID:` An integer uniquely identifying the employee
3. `Salary:` A float representing the employee's salary
4. `Department:` A string (up to 50 characters) representing the department to which the employee belongs
5. `Years of Experience:` An integer indicating the number of years the employee has worked
6. `Performance Score:` A float, calculated based on a custom formula that considers Salary and Years of Experience

Specifications

1. **Define a Structure:** Define a structure Employee to store the above data fields

2. **Functions to Implement:**
   - `addEmployee`:
     - Dynamically allocates memory for a new employee
     - Prompts the user for input to populate the employee's data and calculates the employee's Performance Score as follows:
       * `Performance Score = (0.3 * Years of Experience + 0.7 * Salary / 1000)`
       * Store the employee record in a dynamically growing array
   - `updateEmployee`:
     - Allows updating an existing employee's department and salary by searching for the employee using their Employee ID. Recalculate the `Performance Score` after any salary update
   - `displayEmployees`:
     - Displays all employees, sorted in descending order by their Performance Score. Ensure each field is displayed in a formatted manner
   - `searchEmployeeByName`:
     - Allows searching for employees by a substring of their name. Displays all matching records along with their details
   - `sortEmployeesByPerformance`:
     - Sorts the employee records based on Performance Score in descending order. This function should be called whenever `displayEmployees` or `searchEmployeeByName` is invoked
3. **Main Function:**
   - In the `main()` function, display a `menu` that repeatedly offers the following options until the user chooses to exit:
     - **Add Employee:** Calls `addEmployee()` to add a new employee
     - **Update Employee:** Calls `updateEmployee()` to update an existing employee's department and salary
     - **Display All Employees:** Calls `displayEmployees()` to display a sorted list of all employees by their Performance Score
     - **Search Employee by Name:** Calls `searchEmployeeByName()` to search for and display employees based on a substring match in their name
     - **Exit:** Ends the program and frees all dynamically allocated memory to prevent memory leaks

4. **Memory Management:**
   - Use dynamic memory allocation (`malloc`, `realloc`, and `free`) to handle the storage of employee records
   - Ensure all allocated memory is freed at the end of the program

Hints

- Use arrays of structures as you need to store multiple related items (e.g., employee details), where each element in the array holds the same type of data (e.g., employee name, ID, salary)
- If you're passing a dynamically allocated array of structures (like `struct Employee *employees`) to a function and want that function to modify the array (e.g., reallocate more memory), you can pass a pointer to pointer
- Use `fgets` to handle input that may include spaces, for example, the name of an employee here. But remember to clean the input using `strcspn` to replace the trailing newline (`\n`) with the null terminator `\0`. Suggest to explore the `strcspn` function
- To remove a newline (`\n`) left in the buffer after using `fgets` or other input methods, you can process it using `getchar`
- You may use `strstr` function for finding a substring within a string. It's helpful for searches that are not exact but involve finding partial matches (here, searching for employees by a part of their name)

Example

```
Employee Performance Management System
1.  Add Employee
2.  Update Employee
3.  Display All Employees
4.  Search Employee by Name
5.  Exit
Enter your choice:  1

Enter name of employee:  Alice Johnson
Enter employee ID: 1001
Enter salary:  70000.00
Enter department:  HR
Enter years of experience:  5

Employee Performance Management System
1.  Add Employee
```

```
2.  Update Employee
3.  Display All Employees
4.  Search Employee by Name
5.  Exit
Enter your choice:  3

Employee Records (sorted by Performance Score):

Name:  Alice Johnson
Employee ID: 1001
Salary:  70000.00
Department:  HR
Years of Experience:  5
Performance Score:  50.50

Employee Performance Management System
1.  Add Employee
2.  Update Employee
3.  Display All Employees
4.  Search Employee by Name
5.  Exit
Enter your choice:  2
Enter Employee ID to update:  1001
Enter new salary:  75000.00
Enter new department:  Finance
Employee information updated successfully!

Employee Performance Management System
1.  Add Employee
2.  Update Employee
3.  Display All Employees
4.  Search Employee by Name
5.  Exit
Enter your choice:  4
Enter name to search:  Alice

Search Results:

Name:  Alice Johnson
Employee ID: 1001
Salary:  75000.00
```

```
Department:  Finance
Years of Experience:  5
Performance Score:  54.00

Employee Performance Management System
1.  Add Employee
2.  Update Employee
3.  Display All Employees
4.  Search Employee by Name
5.  Exit
Enter your choice:  5
Exiting...
```