C Coding Question

PART-1 (File Handling and Dynamic Memory Allocation)

Define a structure named `Address` with the following members:

- `street`: character array of size 50
- `city`: character array of size 30
- `zipCode`: integer

Define another structure named `Employee` that contains the following:

- `name`: character array of size 50
- `id`: integer
- `department`: character array of size 30
- `salary`: float
- `address`: a nested `Address` structure

Implement the following functions:

1. `void readAddress(struct Address* a)`: Reads address details (street, city, and zip code) from the user.
2. `void printAddress(const struct Address* a)`: Prints the address details.
3. `void readEmployee(struct Employee* e)`: Reads employee details, including their address (call `readAddress` inside this function).
4. `void printEmployee(const struct Employee* e)`: Prints the employee details, including their address (call `printAddress` inside this function).
5. `float calculateAverageSalary(struct Employee* employees, int count)`: Calculates and returns the average salary of all employees.

In the `main` function:

1. Ask the user to enter how many employees they want to register.
2. Dynamically allocate memory for the employees.
3. Read the details of all employees using `readEmployee`.
4. Print the details of all employees using `printEmployee`.
5. Calculate and display the average salary using `calculateAverageSalary`.
6. Write all employee details to a binary file named `"employees.dat"`.

PART-2 (File Handling)

Implement the following functionality:

1. Read all employee details back from the `"employees.dat"` file and display them using `printEmployee`.
2. Implement a function `struct Employee* searchEmployeeById(struct Employee* employees, int count, int id)` that searches for an employee by their ID and returns a pointer to the `Employee` structure if found, or `NULL` if not.
3. After reading the employees from the file, prompt the user to search for an employee by ID and display their details if found, or an appropriate message if not.

Constraints and Validation:

1. The salary must be a positive number.
2. The employee ID must be unique while reading data.
3. The zip code must be a 5-digit number.

Bonus Challenge: Add a function `void updateEmployeeSalary(struct Employee* e, float newSalary)` that allows the user to update the salary of an employee found using `searchEmployeeById`, and save the updated data back to the file.