Project Description: Employee Management System

Overview:

This project implements an Employee Management System using a Binary Search Tree (BST) data structure in C. The system allows users to perform various operations such as adding new employees, removing existing employees, searching for employees by ID, displaying all employee details, and computing the average salary of all employees.

Features:

1. **Employee Struct and BST Operations:**
   - **Struct Definition (`struct Employee`)**: Stores details of each employee including ID, name, department, age, salary, and pointers to left and right nodes.
   - **Operations**:
     - **Create Employee (`createEmployee`)**: Allocates memory for a new employee node and initializes its attributes.
     - **Insert Employee (`insertEmployee`)**: Inserts a new employee node into the BST based on the employee ID.
     - **Search Employee (`searchEmployee`)**: Searches for an employee in the BST based on the employee ID.
     - **Remove Employee (`removeEmployee`)**: Removes an employee from the BST while maintaining the BST properties.
     - **Display Employee (`displayEmployee`)**: Displays details of a specific employee.
     - **Display All Employees (`displayAllEmployees`)**: Traverses the BST to display details of all employees.
     - **Compute Average Salary (`computeAverageSalary`)**: Computes the average salary of all employees stored in the BST.
2. **User Interface:**
   - **Menu-Driven Interface**: Provides a user-friendly menu for interacting with the Employee Management System.
   - **Input Handling**: Uses `scanf` for user input to perform operations such as adding, removing, searching, displaying, and computing average salary.
3. **Data Management:**
   - **BST Management**: Utilizes BST operations to efficiently manage and retrieve employee data based on employee IDs.
   - **Memory Management**: Allocates and deallocates memory dynamically using `malloc` and `free` to manage employee nodes.
4. **Error Handling:**
   - **Input Validation**: Ensures valid user inputs for operations requiring numeric IDs and salaries.
   - **Error Messages**: Provides informative error messages for cases such as employee not found or invalid menu choices.

Implementation:

- **Programming Language**: C
- **Data Structure**: Binary Search Tree (BST)

- **Functions**: Includes functions for creating, inserting, searching, removing, displaying employees, and computing average salary.
- **Control Structures**: Uses loops (`while`, `do-while`) and conditional statements (`switch`, `if-else`) to manage program flow and user interactions.
- **File Includes**: Standard C libraries (`stdio.h`, `stdlib.h`, `string.h`) for input/output operations, memory allocation, and string handling.

Purpose:

This project serves as a practical demonstration of using BSTs for managing structured data in a real-world application scenario. It emphasizes efficient data storage, retrieval, and management of employee records, catering to organizational needs for employee data management and analysis.

Future Enhancements:

- **User Authentication**: Implement user login functionality for secure access to employee data.
- **Data Persistence**: Integrate file handling to save and load employee data from external files.
- **Graphical User Interface (GUI)**: Upgrade to a GUI-based interface for improved user interaction and visualization of employee data.

Conclusion:

The Employee Management System in C demonstrates foundational concepts in data structures and programming practices, showcasing how BSTs can be utilized effectively for organizing and managing employee information in a structured manner. It serves as a starting point for building more sophisticated applications in employee management and related domains.