Employee Management System - C Program

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Employee structure
struct Employee {
    int id;
    char name[100];
    char designation[100];
    float salary;
};

// Employee Management System structure
struct EmployeeManagementSystem {
    struct Employee employees[100];
    int employeeCount;
};

// Function to flush the input buffer
void flushInputBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}

// Function to add a new employee
```

```
void addEmployee(struct EmployeeManagementSystem *ems) {
    struct Employee newEmployee;

    printf("Enter ID of the new employee: ");
    while (scanf("%d", &newEmployee.id) != 1) {
        printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    printf("Enter name of the new employee: ");
    fgets(newEmployee.name, sizeof(newEmployee.name), stdin);
    newEmployee.name[strcspn(newEmployee.name, "\n")] = '\0';

    printf("Enter designation of the new employee: ");
    fgets(newEmployee.designation, sizeof(newEmployee.designation), stdin);
    newEmployee.designation[strcspn(newEmployee.designation, "\n")] = '\0';

    printf("Enter salary of the new employee: ");
    while (scanf("%f", &newEmployee.salary) != 1) {
        printf("Invalid input. Please enter a valid number for the salary: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    ems->employees[ems->employeeCount] = newEmployee;
```

```
    ems->employeeCount++;
    printf("New employee added with ID %d\n", newEmployee.id);
}

// Function to set salary for an employee
void setSalary(struct EmployeeManagementSystem *ems) {
    int id;
    float newSalary;

    printf("Enter ID of the employee to set salary: ");
    while (scanf("%d", &id) != 1) {
        printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    printf("Enter new salary of the employee: ");
    while (scanf("%f", &newSalary) != 1) {
        printf("Invalid input. Please enter a valid number for the salary: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    for (int i = 0; i < ems->employeeCount; i++) {
        if (ems->employees[i].id == id) {
            ems->employees[i].salary = newSalary;
```

```
            printf("Salary updated for employee with ID %d\n", id);
            return;
        }
    }
    printf("Employee with ID %d not found in the system.\n", id);
}

// Function to update an employee's details
void updateEmployee(struct EmployeeManagementSystem *ems) {
    int id;
    printf("Enter ID of the employee to update: ");
    while (scanf("%d", &id) != 1) {
        printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    for (int i = 0; i < ems->employeeCount; i++) {
        if (ems->employees[i].id == id) {
            char newName[100] = "";
            char newDesignation[100] = "";
            char salaryInput[20] = "";
            float newSalary = -1.0f;

            printf("Enter new name of the employee (leave blank to keep current): ");
            fgets(newName, sizeof(newName), stdin);
```

```
        newName[strcspn(newName, "\n")] = '\0';

        printf("Enter new designation of the employee (leave blank to keep current): ");
        fgets(newDesignation, sizeof(newDesignation), stdin);
        newDesignation[strcspn(newDesignation, "\n")] = '\0';

        printf("Enter new salary of the employee (leave blank to keep current): ");
        fgets(salaryInput, sizeof(salaryInput), stdin);
        if (strlen(salaryInput) > 1) {
            sscanf(salaryInput, "%f", &newSalary);
        }

        if (strlen(newName) > 0) {
            strcpy(ems->employees[i].name, newName);
        }
        if (strlen(newDesignation) > 0) {
            strcpy(ems->employees[i].designation, newDesignation);
        }
        if (newSalary > 0) {
            ems->employees[i].salary = newSalary;
        }
        printf("Details updated for employee with ID %d\n", id);
        return;
    }
}
printf("Employee with ID %d not found in the system.\n", id);
```

```
}

// Function to remove an employee
void removeEmployee(struct EmployeeManagementSystem *ems) {
    int id;
    printf("Enter ID of the employee to remove: ");
    while (scanf("%d", &id) != 1) {
        printf("Invalid input. Please enter a valid integer for the ID: ");
        flushInputBuffer();
    }
    flushInputBuffer();

    for (int i = 0; i < ems->employeeCount; i++) {
        if (ems->employees[i].id == id) {
            for (int j = i; j < ems->employeeCount - 1; j++) {
                ems->employees[j] = ems->employees[j + 1];
            }
            ems->employeeCount--;
            printf("Employee with ID %d has been removed from the system.\n", id);
            return;
        }
    }
    printf("Employee with ID %d not found in the system.\n", id);
}

// Function to display employees by designation
```

```
void displayEmployeesByDesignation(struct EmployeeManagementSystem *ems) {
    char designation[100];
    printf("Enter designation to search: ");
    fgets(designation, sizeof(designation), stdin);
    designation[strcspn(designation, "\n")] = '\0';

    int found = 0;
    for (int i = 0; i < ems->employeeCount; i++) {
        if (strcmp(ems->employees[i].designation, designation) == 0) {
            printf("ID: %d, Name: %s, Designation: %s, Salary: %.2f\n",
                   ems->employees[i].id, ems->employees[i].name,
                   ems->employees[i].designation, ems->employees[i].salary);
            found = 1;
        }
    }
    if (!found) {
        printf("No employees found with designation %s\n", designation);
    }
}

// Function to search an employee by ID
void searchEmployeeById(struct EmployeeManagementSystem *ems) {
    int id;
    printf("Enter ID of the employee to search: ");
    while (scanf("%d", &id) != 1) {
        printf("Invalid input. Please enter a valid integer for the ID: ");
```

```
        flushInputBuffer();
    }
    flushInputBuffer();

    for (int i = 0; i < ems->employeeCount; i++) {
        if (ems->employees[i].id == id) {
            printf("Employee found:\n");
            printf("ID: %d, Name: %s, Designation: %s, Salary: %.2f\n",
                   ems->employees[i].id, ems->employees[i].name,
                   ems->employees[i].designation, ems->employees[i].salary);
            return;
        }
    }
    printf("Employee with ID %d not found in the system.\n", id);
}

// Main function
int main() {
    struct EmployeeManagementSystem ems;
    ems.employeeCount = 0;
    int choice;

    while (1) {
        printf("\nEmployee Management System Menu:\n");
        printf("1. Add Employee\n");
        printf("2. Update Employee\n");
```

```
printf("3. Remove Employee\n");
printf("4. Set Salary\n");
printf("5. Display Employees by Designation\n");
printf("6. Search Employee by ID\n");
printf("7. Exit\n");

printf("Enter your choice: ");
while (scanf("%d", &choice) != 1) {
    printf("Invalid input. Please enter a valid integer for the choice: ");
    flushInputBuffer();
}
flushInputBuffer();

switch (choice) {
    case 1:
        addEmployee(&ems);
        break;
    case 2:
        updateEmployee(&ems);
        break;
    case 3:
        removeEmployee(&ems);
        break;
    case 4:
        setSalary(&ems);
        break;
```

```
            case 5:
                displayEmployeesByDesignation(&ems);
                break;
            case 6:
                searchEmployeeById(&ems);
                break;
            case 7:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```