MINI PROJECT

EMPLOYEE RECORD SYSTEM

NAME :

CLASS :

REG NO :

SUBJECT :

DATE :

EXPLANATION :

The Employee Record System is a simple C program that uses file handling to store and manage employee data permanently. It uses a structure to hold details such as employee ID, name, and salary, and provides a menu-driven interface to add, display, search, update, and delete records. File operations such as reading, writing, and using a temporary file ensure that data is managed safely and efficiently. This project demonstrates the practical use of C file handling and shows how real-world data can be stored and modified even after the program ends.

ALGORITHM :

```
1. Start
      Define structure Employee with fields (id, name,
     dept, salary).
      Open main menu loop.
2. Display Menu
          Add Employee
          2. Display Employees
          3. Search Employee
          4. Update Employee
          5. Delete Employee
          6. Exit
      3. Add Employee
        Open file in append mode.
        Read employee details from user.
        Write record into file.
        Close file.

  4. Display All Employees
          Open file in read mode.
          Read each record until EOF.
          Display details.
          Close file.

  5. Search Employee by ID
        Open file in read mode.
        Read target ID from user.
        Scan file:
           If ID matches    display details.
          Close file.
```

6. Update Employee

    Open original file (read).
    Open temp file (write).
    Read ID to update.
    Copy each record:
      o If ID matches   read new data   write updated record.
      o Else   write original record.
    Replace original file with temp file.

7. Delete Employee

    Open original file (read).
    Open temp file (write).
    Read ID to delete.
    Copy all records except matching ID.
    Replace original file with temp file.

8. Exit

    Terminate program.

CODE :

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

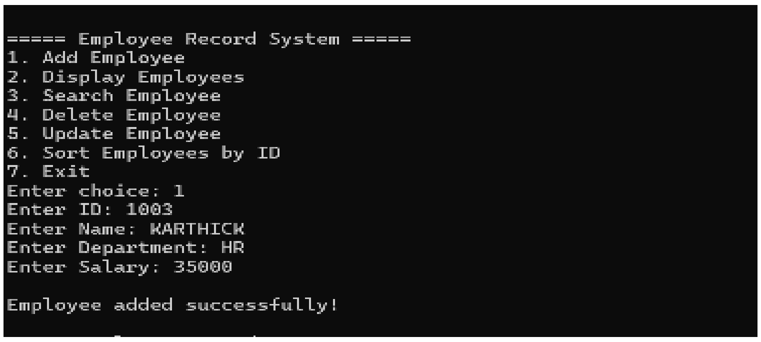
```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define FILENAME "employees.dat"
#define MAX 100
typedef struct {
   int id;
   char name[50];
   char department[50];
   char salary[200];     // SALARY AS STRING
} Employee;

void addEmployee();
void displayEmployees();
void searchEmployee();
void deleteEmployee();
void updateEmployee();
void sortEmployees();
int loadEmployees(Employee emp[], int *size);
void saveEmployees(Employee emp[], int size);
/* -------------------- INPUT VALIDATION
```

```
-------------------- */
int inputInt() {
    char buf[100];
    int valid;
    while (1) {
        scanf("%s", buf);
        valid = 1;
        for (int i = 0; buf[i] != '\0'; i++) {
            if (!isdigit(buf[i])) {
                valid = 0;
                break;
            }
        }
        if (valid) return atoi(buf);
        printf("Error: Numbers only! Enter again: ");
    }
}
void inputString(char *str) {
    int valid;
    while (1) {
        scanf(" %[^\n]s", str);
        valid = 1;
        for (int i = 0; str[i] != '\0'; i++) {
            if (!(isalpha(str[i]) || str[i] == ' ' || str[i] == '.')) {
                valid = 0;
                break;
            }
        }
        if (valid) return;
        printf("Error: Only letters/space allowed! Enter
again: ");
    }
}
void inputSalary(char *str) {
char buf[500];
    int valid, dotCount;
    while (1) {
        scanf("%s", buf);
        valid = 1;
        dotCount = 0;
        for (int i = 0; buf[i] != '\0'; i++) {
            if (buf[i] == '.') {
                dotCount++;
                if (dotCount > 1) { valid = 0; break; }
            }
            else if (!isdigit(buf[i])) {
                valid = 0;
                break;
            }
        }
        if (valid) {
            strcpy(str, buf);
            return;
        }
        printf("Error: Salary must be numeric! Enter again:
");
    }
}
/* -------------------- FILE FUNCTIONS --------------------
*/
int loadEmployees(Employee emp[], int *size) {
    FILE *fp = fopen(FILENAME, "rb");
    if (!fp) {
        *size = 0;
        return 0;
    }
    *size = fread(emp, sizeof(Employee), MAX, fp);
    fclose(fp);
    return *size;
}
void saveEmployees(Employee emp[], int size) {
    FILE *fp = fopen(FILENAME, "wb");
    if (!fp) {
        printf("Error opening file!\n");
        return;
    }
    fwrite(emp, sizeof(Employee), size, fp);
    fclose(fp);
}

/* -------------------- MAIN FEATURES --------------------
*/
void addEmployee() {
    Employee emp[MAX];
    int size;
    loadEmployees(emp, &size);
    Employee e;
    printf("Enter ID: ");
    e.id = inputInt();
    for (int i = 0; i < size; i++) {
        if (emp[i].id == e.id) {
            printf("Error: Employee with ID %d already exists!
\n", e.id);
            return;
        }
    }
    printf("Enter Name: ");
    inputString(e.name);
    printf("Enter Department: ");
    inputString(e.department);
    printf("Enter Salary: ");
    inputSalary(e.salary);
    emp[size] = e;
    size++;
 saveEmployees(emp, size);
    printf("\nEmployee added successfully!\n");
```

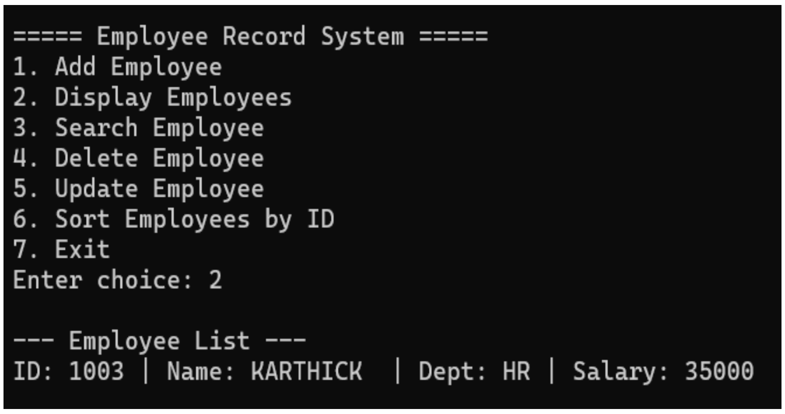
```
}
void displayEmployees() {
    Employee emp[MAX];
    int size;
    loadEmployees(emp, &size);
    if (size == 0) {
        printf("No employees found.\n");
        return;
    }
    printf("\n--- Employee List ---\n");
    for (int i = 0; i < size; i++) {
        printf("ID: %d | Name: %s | Dept: %s | Salary: %s\n",
               emp[i].id, emp[i].name, emp[i].department,
emp[i].salary);
    }
}
void searchEmployee() {
    Employee emp[MAX];
    int size;
    loadEmployees(emp, &size);
```

```
  printf("Enter ID to search: ");
   int id = inputInt();
   for (int i = 0; i < size; i++) {
       if (emp[i].id == id) {
           printf("\nFound Employee:\n");
           printf("ID: %d | Name: %s | Dept: %s | Salary: 
%s\n",
                  emp[i].id, emp[i].name, emp[i].department, 
emp[i].salary);
           return;
       }
   }
   printf("Employee not found.\n");
}
void deleteEmployee() {
   Employee emp[MAX];
   int size;
   loadEmployees(emp, &size);
   printf("Enter ID to delete: ");
   int id = inputInt();
   int found = 0;
   for (int i = 0; i < size; i++) {
       if (emp[i].id == id) {
         for (int j = i; j < size - 1; j++)
               emp[j] = emp[j + 1];
           size--;
           found = 1;
           break;
       }
   }
   if (!found) {
       printf("Employee not found.\n");
       return;
   }
   saveEmployees(emp, size);
   printf("Employee deleted.\n");
}
void updateEmployee() {
   Employee emp[MAX];
   int size;
   loadEmployees(emp, &size);
   printf("Enter ID to update: ");
   int id = inputInt();
   for (int i = 0; i < size; i++) {
       if (emp[i].id == id) {

           printf("Enter new Name: ");
           inputString(emp[i].name);
           printf("Enter new Department: ");
           inputString(emp[i].department);
           printf("Enter new Salary: ");
           inputSalary(emp[i].salary);
           saveEmployees(emp, size);
           printf("Employee updated.\n");
           return;
       }
   }
   printf("Employee not found.\n");
}
void sortEmployees() {
   Employee emp[MAX];
   int size;
   loadEmployees(emp, &size);
   for (int i = 0; i < size - 1; i++) {
       for (int j = i + 1; j < size; j++) {
           if (emp[i].id > emp[j].id) {
               Employee temp = emp[i];
               emp[i] = emp[j];
               emp[j] = temp;
           }
       }
   }
   saveEmployees(emp, size);
   printf("Employees sorted.\n");
}
/* -------------------- MAIN MENU -------------------- */
int main() {
   int choice;
   while (1) {
       printf("\n===== Employee Record System
=====\n");
       printf("1. Add Employee\n");
       printf("2. Display Employees\n");
       printf("3. Search Employee\n");
       printf("4. Delete Employee\n");
       printf("5. Update Employee\n");
       printf("6. Sort Employees by ID\n");
       printf("7. Exit\n");
       printf("Enter choice: ");

       choice = inputInt();
       switch (choice) {
           case 1: addEmployee(); break;
           case 2: displayEmployees(); break;
           case 3: searchEmployee(); break;
           case 4: deleteEmployee(); break;
           case 5: updateEmployee(); break;
           case 6: sortEmployees(); break;
           case 7: exit(0);
           default: printf("Invalid choice!\n");
       }
   }
}
```

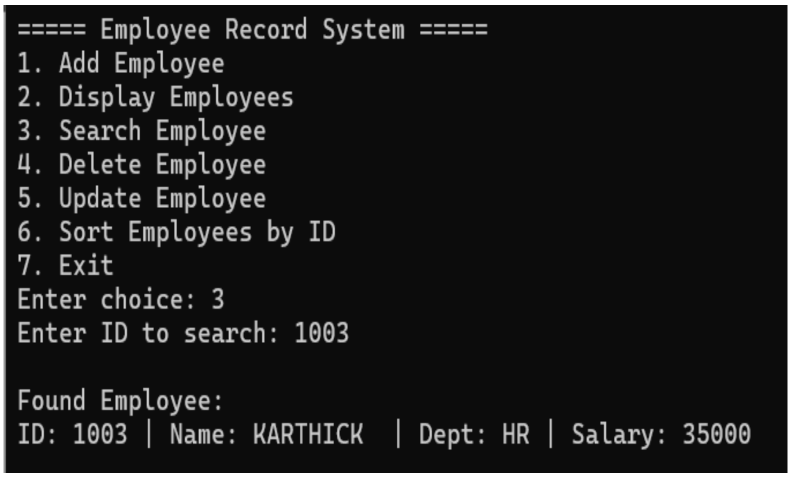
RESULT :



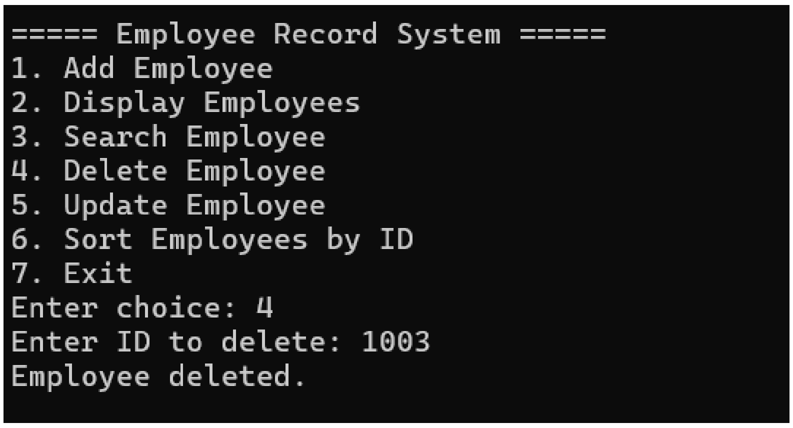
```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 1
Enter ID: 1003
Enter Name: KARTHICK
Enter Department: HR
Enter Salary: 35000

Employee added successfully!
```



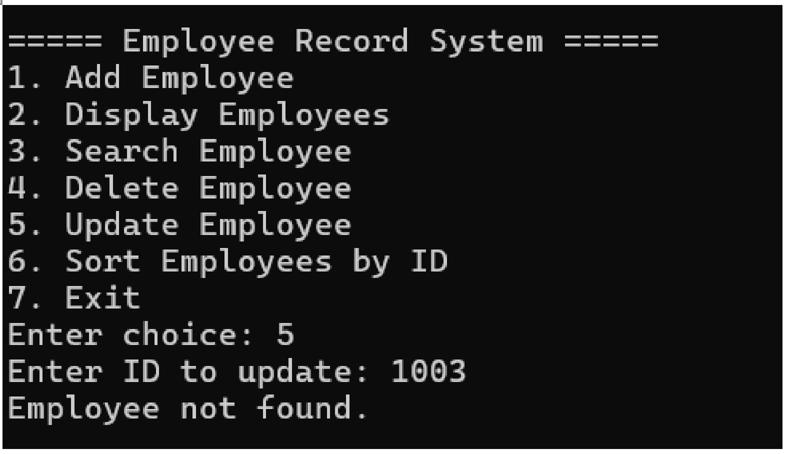
```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 2

--- Employee List ---
ID: 1003 | Name: KARTHICK  | Dept: HR | Salary: 35000
```



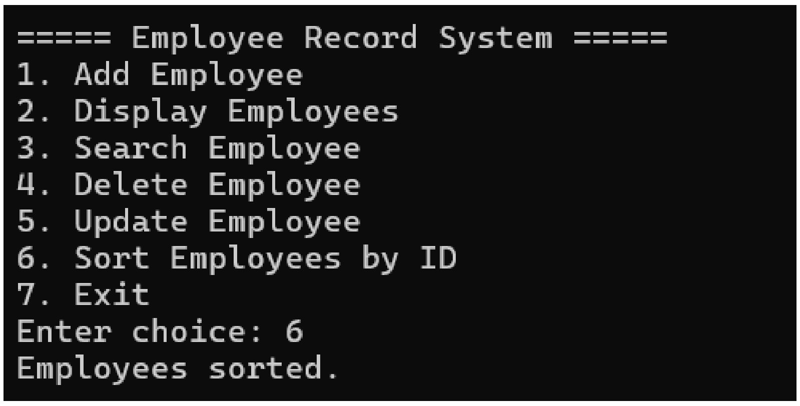
```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 3
Enter ID to search: 1003

Found Employee:
ID: 1003 | Name: KARTHICK  | Dept: HR | Salary: 35000
```



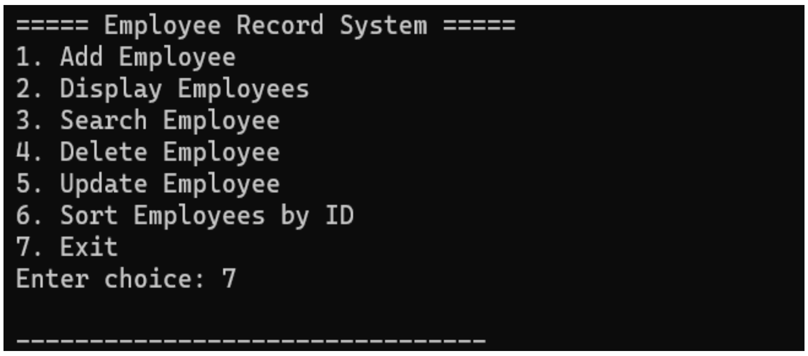
```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 4
Enter ID to delete: 1003
Employee deleted.
```



```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 5
Enter ID to update: 1003
Employee not found.
```



```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 6
Employees sorted.
```



```
===== Employee Record System =====
1. Add Employee
2. Display Employees
3. Search Employee
4. Delete Employee
5. Update Employee
6. Sort Employees by ID
7. Exit
Enter choice: 7

--------------------------------
```