ASSIGNMENT 24

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <iomanip>

using namespace std;

// Structure to hold employee information
struct Employee {
   int id;
   char name[50];
   char designation[50];
   double salary;
};

// Function prototypes
void addEmployee();
void deleteEmployee();
void displayEmployee();

// Global file pointers
fstream dataFile, indexFile;

int main() {
   int choice;
   dataFile.open("employees.dat", ios::in | ios::out | ios::ate | ios::binary);
   indexFile.open("index.dat", ios::in | ios::out | ios::ate | ios::binary);
```

```
do {
    cout << "\n\nEmployee Information System";
    cout << "\n1. Add Employee";
    cout << "\n2. Delete Employee";
    cout << "\n3. Display Employee";
    cout << "\n4. Exit";
    cout << "\nEnter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            addEmployee();
            break;
        case 2:
            deleteEmployee();
            break;
        case 3:
            displayEmployee();
            break;
        case 4:
            cout << "Exiting..." << endl;
            break;
        default:
            cout << "Invalid choice! Try again." << endl;
    }
} while (choice != 4);

dataFile.close();
indexFile.close();

return 0;
```

```
}

// Function to add an employee
void addEmployee() {
    Employee emp;
    int id;
    cout << "\nEnter Employee ID: ";
    cin >> id;
    emp.id = id;

    // Check if employee already exists
    indexFile.seekg(0, ios::beg);
    int indexId, offset;
    while (indexFile.read(reinterpret_cast<char*>(&indexId), sizeof(int))) {
        indexFile.read(reinterpret_cast<char*>(&offset), sizeof(int));
        if (indexId == id) {
            cout << "Employee with ID " << id << " already exists!" << endl;
            return;
        }
    }

    cout << "Enter Employee Name: ";
    cin.ignore();
    cin.getline(emp.name, 50);
    cout << "Enter Designation: ";
    cin.getline(emp.designation, 50);
    cout << "Enter Salary: ";
    cin >> emp.salary;

    // Write employee data to file
    int offset = dataFile.tellp();
```

```
    dataFile.write(reinterpret_cast<char*>(&emp), sizeof(Employee));

    // Write index entry
    indexFile.seekp(0, ios::end);
    indexFile.write(reinterpret_cast<char*>(&id), sizeof(int));
    indexFile.write(reinterpret_cast<char*>(&offset), sizeof(int));

    cout << "Employee added successfully!" << endl;
}

// Function to delete an employee
void deleteEmployee() {
    int id;
    cout << "\nEnter Employee ID to delete: ";
    cin >> id;

    // Check if employee exists
    indexFile.seekg(0, ios::beg);
    int indexId, offset;
    bool found = false;
    while (indexFile.read(reinterpret_cast<char*>(&indexId), sizeof(int))) {
        indexFile.read(reinterpret_cast<char*>(&offset), sizeof(int));
        if (indexId == id) {
            found = true;
            break;
        }
    }

    if (!found) {
        cout << "Employee with ID " << id << " not found!" << endl;
        return;
```

```
    }

    // Delete employee data from file
    dataFile.seekp(offset, ios::beg);
    Employee dummy = {0, "", "", 0.0};
    dataFile.write(reinterpret_cast<char*>(&dummy), sizeof(Employee));

    // Update index file
    indexFile.close();
    indexFile.open("index.dat", ios::out | ios::trunc | ios::binary);
    dataFile.seekg(0, ios::beg);
    Employee emp;
    while (dataFile.read(reinterpret_cast<char*>(&emp), sizeof(Employee))) {
        if (emp.id != 0) {
            int offset = dataFile.tellg() - sizeof(Employee);
            indexFile.write(reinterpret_cast<char*>(&emp.id), sizeof(int));
            indexFile.write(reinterpret_cast<char*>(&offset), sizeof(int));
        }
    }

    cout << "Employee deleted successfully!" << endl;
}

// Function to display an employee
void displayEmployee() {
    int id;
    cout << "\nEnter Employee ID to display: ";
    cin >> id;

    // Check if employee exists
    indexFile.seekg(0, ios::beg);
```

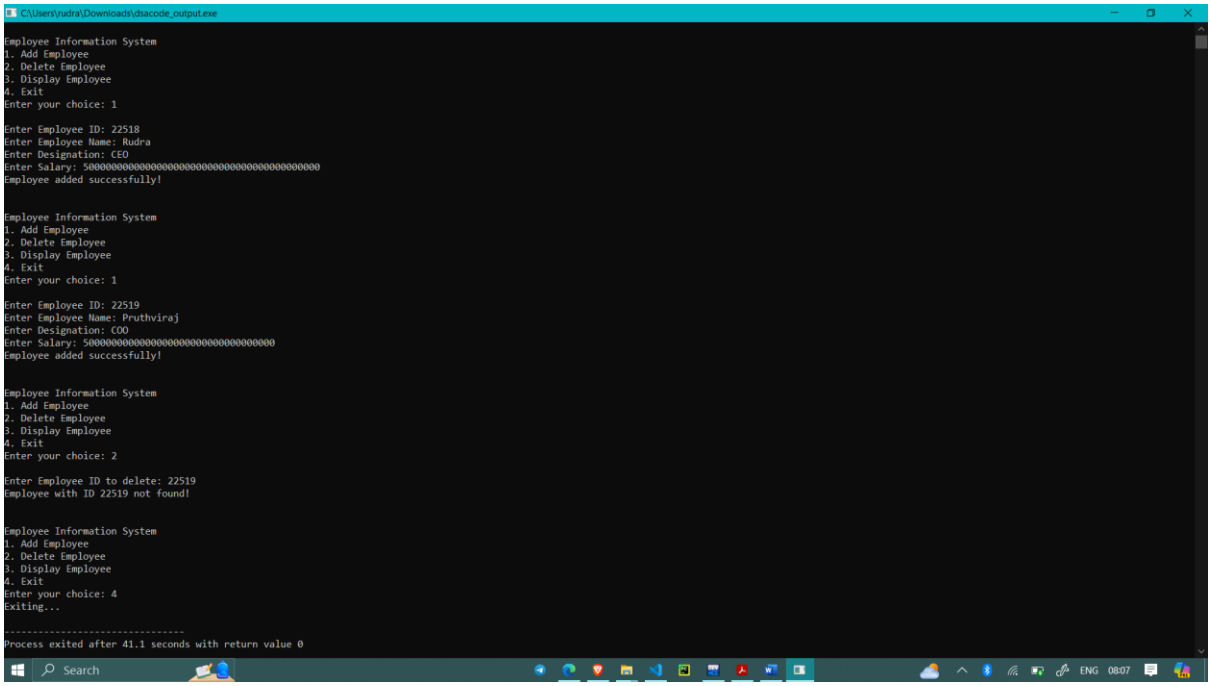
```
    int indexId, offset;
    bool found = false;
    while (indexFile.read(reinterpret_cast<char*>(&indexId), sizeof(int))) {
        indexFile.read(reinterpret_cast<char*>(&offset), sizeof(int));
        if (indexId == id) {
            found = true;
            break;
        }
    }

    if (!found) {
        cout << "Employee with ID " << id << " not found!" << endl;
        return;
    }

    // Display employee details
    dataFile.seekg(offset, ios::beg);
    Employee emp;
    dataFile.read(reinterpret_cast<char*>(&emp), sizeof(Employee));

    cout << "\nEmployee Details:" << endl;
    cout << "ID: " << emp.id << endl;
    cout << "Name: " << emp.name << endl;
    cout << "Designation: " << emp.designation << endl;
    cout << "Salary: " << fixed << setprecision(2) << emp.salary << endl;
}
```

OUTPUT :


C:\Users\rudra\Downloads\dsacode_output.exe
Employee Information System
1. Add Employee
2. Delete Employee
3. Display Employee
4. Exit
Enter your choice: 1
Enter Employee ID: 22518
Enter Employee Name: Rudra
Enter Designation: CEO
Enter Salary: 5000000000000000000000000000000000000000000
Employee added successfully!
Employee Information System
1. Add Employee
2. Delete Employee
3. Display Employee
4. Exit
Enter your choice: 1
Enter Employee ID: 22519
Enter Employee Name: Pruthviraj
Enter Designation: COO
Enter Salary: 500000000000000000000000000000000000
Employee added successfully!
Employee Information System
1. Add Employee
2. Delete Employee
3. Display Employee
4. Exit
Enter your choice: 2
Enter Employee ID to delete: 22519
Employee with ID 22519 not found!
Employee Information System
1. Add Employee
2. Delete Employee
3. Display Employee
4. Exit
Enter your choice: 4
Exiting...
Process exited after 41.1 seconds with return value 0
Search
ENG
08:07