**Institute of Computer Technology**

**B. Tech. Computer Science and Engineering**

**Sub: ESFP – II**

**Course Code: 2CSE203**

**Practical – 15**

**Name: Jaymin Gondaliya**

**Enrollment No: 23162171007**

**Sem - 2**

**Branch: CS**

**Class: B**

**Batch: 25**

**Objective:**

To learn about object-oriented concept like inheritance, template function, template class, and file handling concept..

**Problem Definition-1:** Complete the code for the object assigned to you to satisfy the following specifications.

**Code:**

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

using namespace std;

class Employee {

public:

    string empName;

    int empAge;

    string empDepartment;

public:

    Employee() {}

    Employee(string n, int a, string d) : empName(n), empAge(a), empDepartment(d) {}

    ~Employee() {}

    void displayEmpRecord() {

        cout << "Name: " << empName << endl;

        cout << "Age: " << empAge << endl;

        cout << "Department: " << empDepartment << endl;

    }

};

template<typename T>

class RecordManager {

    T\* records;

    int maxSize;

    int currentSize;

public:

    RecordManager(int size) : maxSize(size), currentSize(0) {

        records = new T[maxSize];

        loadFromFile("employees.txt");

    }

    ~RecordManager() {

        delete[] records;

    }

    void addRecord(const T& record) {

        if (currentSize < maxSize) {

            records[currentSize++] = record;

            cout << "Record added successfully." << endl;

            saveToFile("employees.txt");

        } else {

            cout << "Maximum capacity reached. Cannot add more records." << endl;

        }

    }

    void displayRecords() {

        if (currentSize > 0) {

            for (int i = 0; i < currentSize; ++i) {

                records[i].displayEmpRecord();

            }

        } else {

            cout << "No records available." << endl;

        }

    }

    void searchRecord(const string& searchName) {

        bool found = false;

        for (int i = 0; i < currentSize; ++i) {

            if (records[i].empName == searchName) {

                cout << "Record found:" << endl;

                records[i].displayEmpRecord();

                found = true;

                break;

            }

        }

        if (!found) {

            cout << "Record not found!" << endl;

        }

    }

    void updateRecord(const string& searchName, const T& updatedRecord) {

        bool found = false;

        for (int i = 0; i < currentSize; ++i) {

            if (records[i].empName == searchName) {

                records[i] = updatedRecord;

                cout << "Record updated successfully." << endl;

                saveToFile("employees.txt");

                found = true;

                break;

            }

        }

        if (!found) {

            cout << "Record not found! Cannot update." << endl;

        }

    }

    void deleteAllRecords() {

        currentSize = 0;

        cout << "All records deleted!" << endl;

        saveToFile("employees.txt");

    }

    void saveToFile(const string& filename) {

        ofstream file(filename);

        if (file.is\_open()) {

            for (int i = 0; i < currentSize; ++i) {

                file << records[i].empName << " " << records[i].empAge << " " << records[i].empDepartment << endl;

            }

            file.close();

            cout << "Data saved to file: " << filename << endl;

        } else {

            cout << "Unable to open file: " << filename << endl;

        }

    }

    void loadFromFile(const string& filename) {

        ifstream file(filename);

        if (file.is\_open()) {

            currentSize = 0;

            while (!file.eof()) {

                string name, department;

                int age;

                file >> name >> age >> department;

                if (name != "") {

                    records[currentSize++] = T(name, age, department);

                }

            }

            file.close();

            cout << "Data loaded from file: " << filename << endl;

        } else {

            ofstream newFile(filename);

            if (newFile.is\_open()) {

                newFile.close();

                cout << "File created: " << filename << endl;

            } else {

                cout << "Unable to create file: " << filename << endl;

            }

        }

    }

};

int main() {

    const int MAX\_EMPLOYEES = 5;

    RecordManager<Employee> empManager(MAX\_EMPLOYEES);

    int choice;

    do {

        cout << "Menu:" << endl;

        cout << "1. Add new employee" << endl;

        cout << "2. Display all employees" << endl;

        cout << "3. Search employee" << endl;

        cout << "4. Update employee" << endl;

        cout << "5. Delete all employees" << endl;

        cout << "6. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1: {

                string name, department;

                int age;

                cout << "Enter details for new employee:" << endl;

                cout << "Name: ";

                cin >> name;

                cout << "Age: ";

                cin >> age;

                cout << "Department: ";

                cin >> department;

                empManager.addRecord(Employee(name, age, department));

                break;

            }

            case 2:

                empManager.displayRecords();

                break;

            case 3: {

                string searchName;

                cout << "Enter name to search: ";

                cin >> searchName;

                empManager.searchRecord(searchName);

                break;

            }

            case 4: {

                string searchName, newName, newDepartment;

                int newAge;

                cout << "Enter name to update: ";

                cin >> searchName;

                cout << "Enter new details:" << endl;

                cout << "Name: ";

                cin >> newName;

                cout << "Age: ";

                cin >> newAge;

                cout << "Department: ";

                cin >> newDepartment;

                empManager.updateRecord(searchName, Employee(newName, newAge, newDepartment));

                break;

            }

            case 5:

                empManager.deleteAllRecords();

                break;

            case 6:

                cout << "Exiting program." << endl;

                break;

            default:

                cout << "Invalid choice. Please enter a number between 1 and 6." << endl;

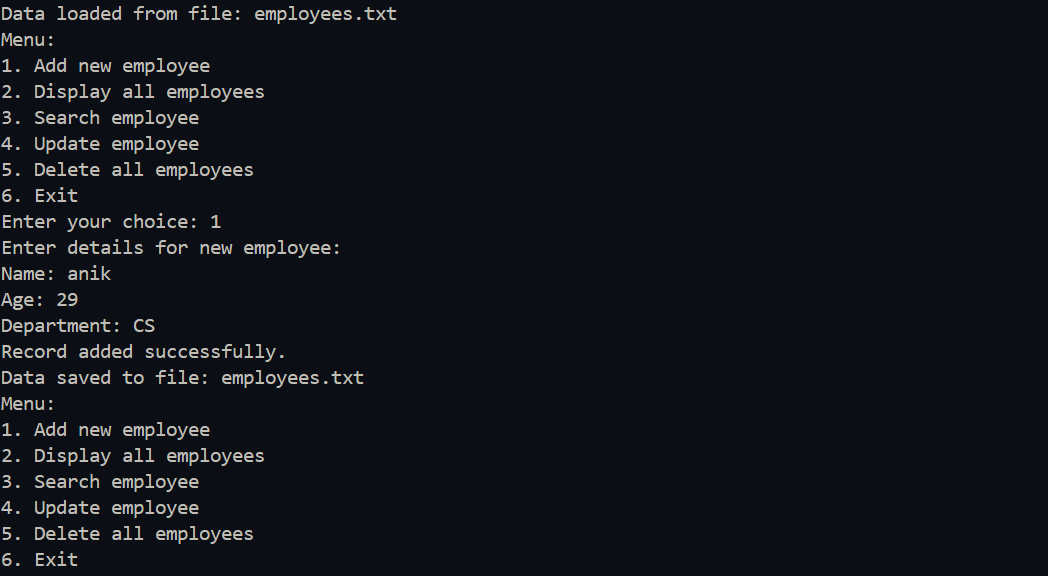
        }

    } while (choice != 6);

    return 0;

}

**Output:**

****

****