/\*

Create a program that models a College Management System using nested structures.

The system should manage information about departments, professors, and students.

Requirements:

Define Structures:

Student:

rollNumber (integer): Unique ID for the student.

name (string): Full name of the student.

marks (array of 5 floats): Marks in 5 subjects.

percentage (float): Calculated based on marks.

Professor:

profID (integer): Unique ID for the professor.

name (string): Full name of the professor.

subject (string): Subject taught by the professor.

Department:

deptID (integer): Unique ID for the department.

name (string): Name of the department.

professors (array of Professor): List of professors in the department.

students (array of Student): List of students in the department.

Features:

Add Departments:

Dynamically create a new department with associated professors and students.

Display Details:

Recursively display all details of departments, including associated professors and students.

Calculate Percentage:

For each student, calculate their percentage from their marks.

Search:

Search for a professor by profID and display their details.

Search for a student by rollNumber and display their details.

Update Records:

Update a student's marks and recalculate their percentage.

Update a professor's subject.

Example Menu:

Menu:

1. Add a New Department

2. Add Professors to a Department

3. Add Students to a Department

4. Display Department Details

5. Search for a Professor by ID

6. Search for a Student by Roll Number

7. Update Student Marks

8. Update Professor's Subject

9. Exit

Sample Scenario:

Input:

Add a department "Computer Science" with ID 101.

Add 2 professors: Prof. Alice (teaching "Data Structures") and Prof. Bob (teaching "Algorithms").

Add 3 students: John, Emma, and Mike with marks in 5 subjects.

Output:

Display department details:

Department ID: 101

Department Name: Computer Science

Professors:

1. ProfID: 1, Name: Alice, Subject: Data Structures

2. ProfID: 2, Name: Bob, Subject: Algorithms

Students:

1. RollNumber: 1, Name: John, Percentage: 85.6%

2. RollNumber: 2, Name: Emma, Percentage: 92.4%

3. RollNumber: 3, Name: Mike, Percentage: 78.5%

Search:

Search for student with RollNumber 2:

Found Student:

RollNumber: 2, Name: Emma, Percentage: 92.4%

has context menu\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Student

{

    int rollNumber;

    char name[50];

    float marks[5];

    float percentage;

};

struct Professor

{

    int profID;

    char name[30];

    char subject[20];

};

struct Department

{

    int deptID;

    char name[20];

    struct Professor p[10];

    struct Student s[10];

    int profCount;

    int studCount;

};

struct Department \*departments=NULL;

int deptCount=0;

void addDepartment();

void addProfessor();

void addStudent();

void displayDepartments();

void searchProfByID();

void searchStudByNum();

void updateStudMarks();

void updateProfSub();

int main()

{

    int choice;

    while (1)

    {

        printf("\nMenu:\n");

        printf("1. Add a New Department\n2. Add Professors to a Department\n3. Add Students to a Department\n4. Display Department Details\n5. Search for a Professor by ID\n6. Search for a Student by Roll Number\n7. Update Student Marks\n8. Update Professor's Subject\n9. Exit\n");

        printf("Enter your choice: ");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

                addDepartment();

                break;

            case 2:

                addProfessor();

                break;

            case 3:

                addStudent();

                break;

            case 4:

                displayDepartments();

                break;

            case 5:

                searchProfByID();

                 break;

            case 6:

                searchStudByNum();

                break;

            case 7:

                updateStudMarks();

                break;

            case 8:

                updateProfSub();

                break;

            case 9:

                exit(0);

            default:

                printf("Invalid choice. Try again.\n");

                break;

        }

    }

    return 0;

}

void addDepartment()

{

    deptCount++;

    departments = realloc(departments,deptCount\*sizeof(struct Department));

    struct Department \*dept = &departments[deptCount-1];

    printf("Enter department ID: ");

    scanf("%d",&dept->deptID);

    printf("Enter department name: ");

    scanf("%s",dept->name);

    dept->profCount=0;

    dept->studCount=0;

    printf("Department added successfully.\n");

}

void addProfessor()

{

    int deptID;

    printf("enter department ID: ");

    scanf("%d",&deptID);

    struct Department \*dept=NULL;

    for(int i=0;i<deptCount;i++)

    {

        if(departments[i].deptID==deptID)

        {

            dept =&departments[i];

            break;

        }

    }

    if(!dept)

    {

        printf("department not found.\n");

        return;

    }

    struct Professor \*prof =&dept->p[dept->profCount++];

    printf("enter professor ID: ");

    scanf("%d",&prof->profID);

    printf("enter professor name: ");

    scanf("%s",prof->name);

    printf("enter subject: ");

    scanf("%s",prof->subject);

    printf("professor added successfully.\n");

}

void addStudent()

{

    int deptID;

    printf("enter department ID: ");

    scanf("%d",&deptID);

    struct Department \*dept=NULL;

    for(int i=0;i<deptCount;i++)

    {

        if(departments[i].deptID==deptID)

        {

            dept= &departments[i];

            break;

        }

    }

    if(!dept)

    {

        printf("department not found.\n");

        return;

    }

    struct Student \*stud= &dept->s[dept->studCount++];

    printf("Enter student roll number: ");

    scanf("%d",&stud->rollNumber);

    printf("Enter student name: ");

    scanf("%s",stud->name);

    printf("Enter marks in 5 subjects: ");

    for (int i=0;i<5;i++)

    {

        scanf("%f",&stud->marks[i]);

    }

    float total=0;

    for (int i=0;i<5;i++)

    {

        total+=stud->marks[i];

    }

    stud->percentage =total/5;

    printf("student added successfully.\n");

}

void displayDepartments()

{

    if (deptCount==0)

    {

        printf("no departments to display.\n");

        return;

    }

    for (int i=0;i<deptCount;i++)

    {

        struct Department \*dept = &departments[i];

        printf("\ndepartment ID: %d\n",dept->deptID);

        printf("department Name: %s\n",dept->name);

        printf("professors:\n");

        for (int j=0;j<dept->profCount;j++)

        {

            struct Professor \*prof = &dept->p[j];

            printf("  ID: %d, Name: %s, subject: %s\n",prof->profID,prof->name,prof->subject);

        }

        printf("students:\n");

        for (int j=0;j<dept->studCount;j++)

        {

            struct Student \*stud = &dept->s[j];

            printf("  roll Number: %d, name: %s, percentage: %.2f\n",stud->rollNumber,stud->name,stud->percentage);

        }

    }

}

void searchProfByID()

{

    int profID;

    printf("enter professor ID: ");

    scanf("%d",&profID);

    for (int i = 0; i < deptCount; i++)

    {

        struct Department \*dept = &departments[i];

        for (int j=0;j<dept->profCount;j++)

        {

            struct Professor \*prof = &dept->p[j];

            if (prof->profID == profID)

            {

                printf("professor found in department %s:\n",dept->name);

                printf("  ID: %d, name: %s, subject: %s\n",prof->profID,prof->name,prof->subject);

                return;

            }

        }

    }

    printf("Professor not found.\n");

}

void searchStudByNum()

{

    int rollNumber;

    printf("enter student roll number: ");

    scanf("%d",&rollNumber);

    for (int i = 0;i<deptCount;i++)

    {

        struct Department \*dept =&departments[i];

        for (int j = 0;j<dept->studCount;j++)

        {

            struct Student \*stud=&dept->s[j];

            if (stud->rollNumber == rollNumber)

            {

                printf("student found in department %s:\n", dept->name);

                printf("  roll Number: %d, name: %s, percentage: %.2f\n",

                       stud->rollNumber, stud->name, stud->percentage);

                return;

            }

        }

    }

    printf("student not found.\n");

}

void updateStudMarks()

{

    int rollNumber;

    printf("enter student roll number: ");

    scanf("%d", &rollNumber);

    for (int i=0;i<deptCount;i++)

    {

        struct Department \*dept = &departments[i];

        for (int j=0;j<dept->studCount;j++)

        {

            struct Student \*stud = &dept->s[j];

            if (stud->rollNumber==rollNumber)

            {

                printf("enter new marks in 5 subjects: ");

                for (int k=0;k<5;k++)

                {

                    scanf("%f",&stud->marks[k]);

                }

                float total =0;

                for (int k=0;k<5;k++)

                {

                    total+= stud->marks[k];

                }

                stud->percentage=total/5;

                printf("student marks updated successfully.\n");

                return;

            }

        }

    }

    printf("student not found.\n");

}

void updateProfSub()

{

    int profID;

    printf("enter professor ID: ");

    scanf("%d", &profID);

    for (int i=0;i<deptCount;i++)

    {

        struct Department \*dept = &departments[i];

        for (int j=0;j<dept->profCount;j++)

        {

            struct Professor \*prof = &dept->p[j];

            if (prof->profID == profID)

            {

                printf("enter new subject: ");

                scanf("%s",prof->subject);

                printf("professor subject updated successfully.\n");

                return;

            }

        }

    }

}