ASSIGNMENT 1:

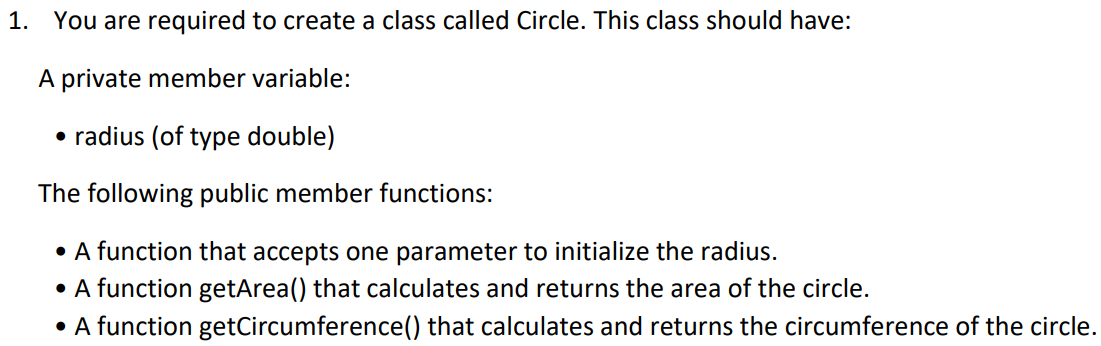
Name: Om Prasanna Kulkarni

PRN: 22510034

Batch: S3

Branch: CSE

Topic: Basics of Object Oriented Programming



#include<iostream>

using namespace std;

#define pi 3.14

class Circle{

  float r;

  public:

   void acceptParam();

   float getArea();

   float getCircumference();

};

void Circle::acceptParam(){

    cout<<"Enter the radius of the circle to be operated on: "<<endl;

    cin>>r;

}

float Circle::getArea(){

    float area;

    area =pi\*r\*r;

    return area;

}

float Circle::getCircumference(){

    float circumference;

    circumference=2\*pi\*r;

    return circumference;

}

int main(){

    Circle a;

    a.acceptParam();

 float  b= a.getArea();

 float  c= a.getCircumference();

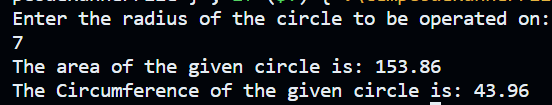
    cout<<"The area of the given circle is: "<<b<<endl;

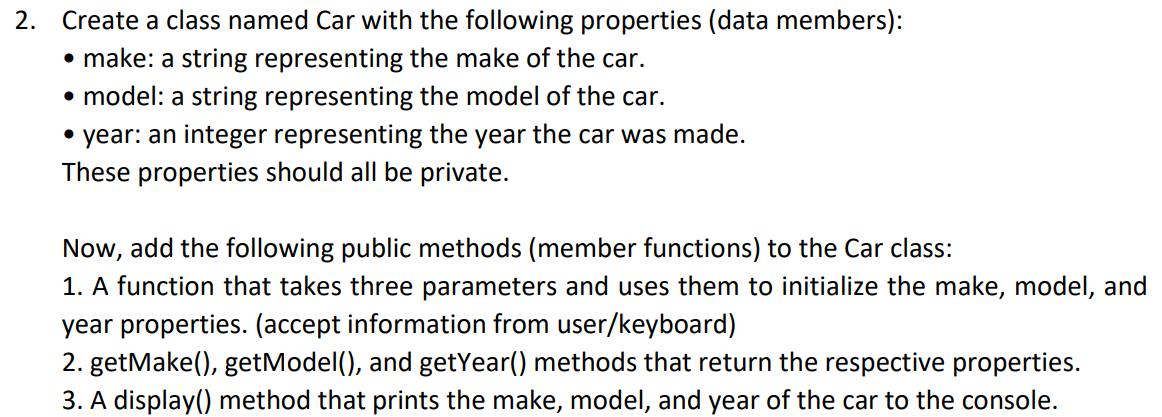
    cout<<"The Circumference of the given circle is: "<<c<<endl;

    return 0;

}

OUTPUT:





#include<iostream>

using namespace std;

class Car{

    string makecar,modelcar;

    int year;

    public:

    void initializingCar(string makecar,string modelcar,int year){

        makecar=makecar;

        modelcar=modelcar;

        year=year;

    }

    void initializingCar();

    string getMake();

    string getModel();

    int getYear();

    void displayCar();

};

void Car::displayCar(){

    Car p;

    string x=p.getMake();

    string y=p.getModel();

    int z=p.getYear();

    cout<<endl;

    cout<<"The make of the Car:"<<x<<endl;

    cout<<"The model of the car:"<<y<<endl;

    cout<<"The Year of the launching of the car:"<<z<<endl;

}

string Car::getMake(){

    cout<<"Input the making of the car:"<<endl;

    cin>>makecar;

    string s;

    s=makecar;

    return s;

}

string Car::getModel(){

    cout<<"Input the model of the car:"<<endl;

    cin>>modelcar;

    string s;

    s=modelcar;

    return s;

}

int Car::getYear(){

    cout<<"Input the year in which the car was launched:"<<endl;

    cin>>year;

    return year;

    }

int main(){

    Car a;

    string makecar,modelcar;

    int year;

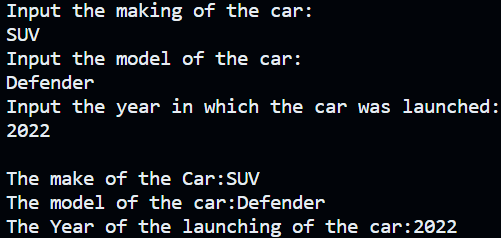
    a.initializingCar(makecar,modelcar,year);

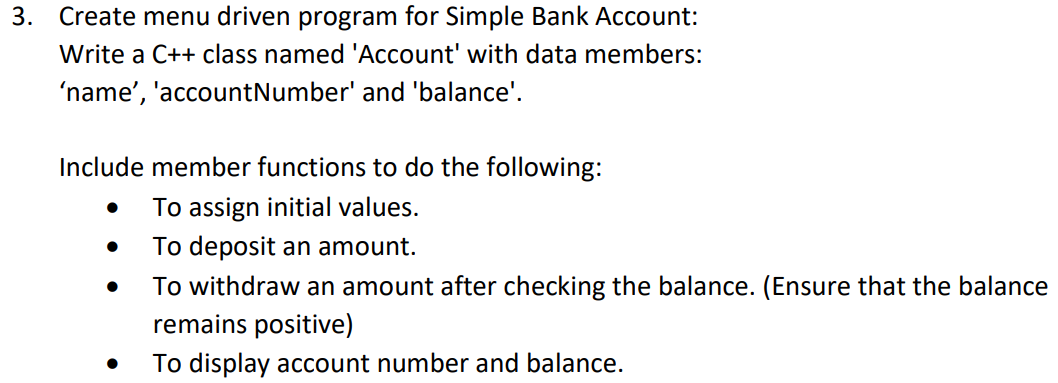
    a.displayCar();

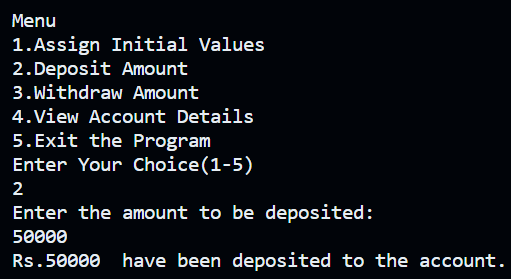
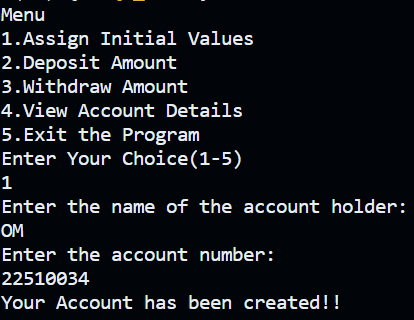
    return 0;

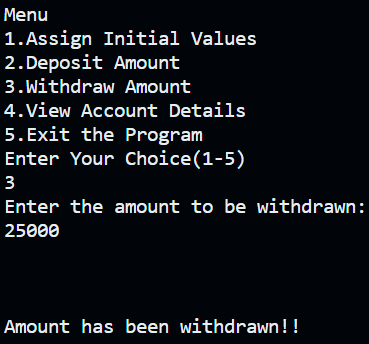
}

OUTPUT:

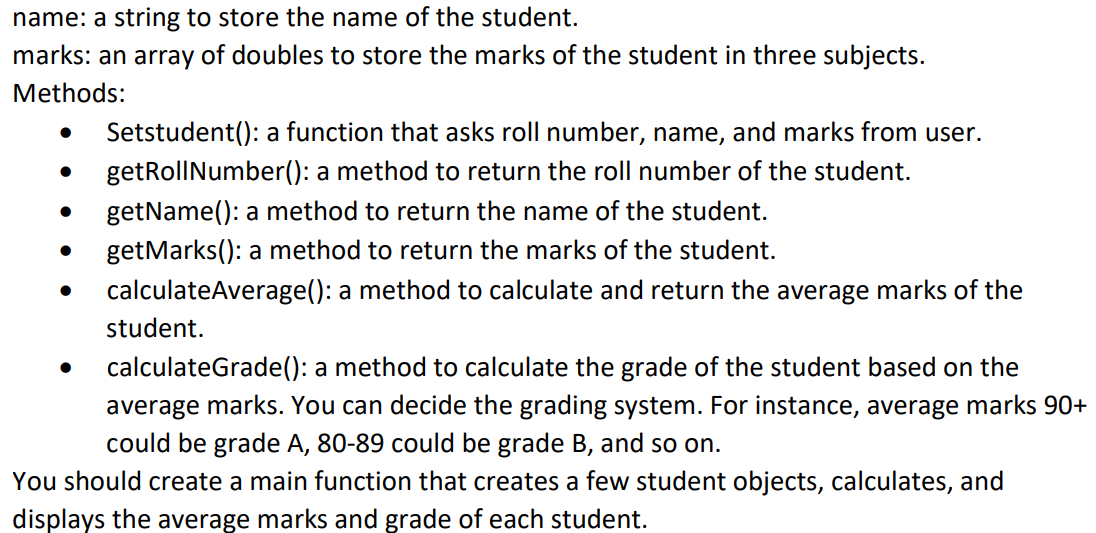








4. Student result generation You need to create a 'Student' class with the following specifications: Attributes: rollNumber: an integer to store the roll number of the student.



#include<iostream>

using namespace std;

class Student{

   int rollNumber;

   string name;

   double marks[3];

public:

void SetStudent();

int getRollNumber();

string getName();

double getMarks();

double calculateAverage();

char calculateGrade();

double average;

};

void Student::SetStudent(){

    cout<<"Enter the Roll Number:\n";

    cin>>rollNumber;

    cin.ignore();

    cout<<"Enter the Name:\n";

    cin.ignore();

    getline(cin,name);

    cout<<"Enter the marks obtained in the 3 subjects:(Out of 100)\n";

    for(int i=0;i<3;i++){cin>>marks[i];}

}

int Student::getRollNumber(){

   return rollNumber;

}

string Student::getName(){

    return name;

}

double Student::getMarks(){

    for(int i=0;i<3;i++){

        return marks[i];

    }

}

double Student::calculateAverage(){

   average=(marks[0]+marks[1]+marks[2])/3;

   return average;

}

char Student::calculateGrade(){

   if(average>=90&&average<=100){

    return 'A';

   }

   else if(average>=80&&average<=90){

    return 'B';

   }

   else if(average>=70&&average<=80){

    return 'C';

   }

   else if(average>=60&&average<=70){

    return 'D';

   }

   else if(average>=50&&average<=60){

    return 'E';

   }

   else{

    return 'FF';

   }

}

int main(){

    int studetncount;

    cout<<"Enter the number of students:\n";

    cin>>studetncount;

    Student obj[studetncount];

    for(int i=0;i<studetncount;i++){

        cout<<"Enter Student Info:"<<i+1<<endl;

        obj[i].SetStudent();

        cout<<endl;

    }

    cout<<"Results:\n";

    for(int i=0;i<studetncount;i++){

        cout<<"Student: "<<i+1<<endl;

        cout<<"Student Roll Number:"<<obj[i].getRollNumber()<<endl;

        cout<<"Name:"<<obj[i].getName()<<endl;

       cout<<"The Average of the student is:"<<obj[i].calculateAverage()<<endl;

       cout<<"The Grade of the student is:"<<obj[i].calculateGrade()<<endl;

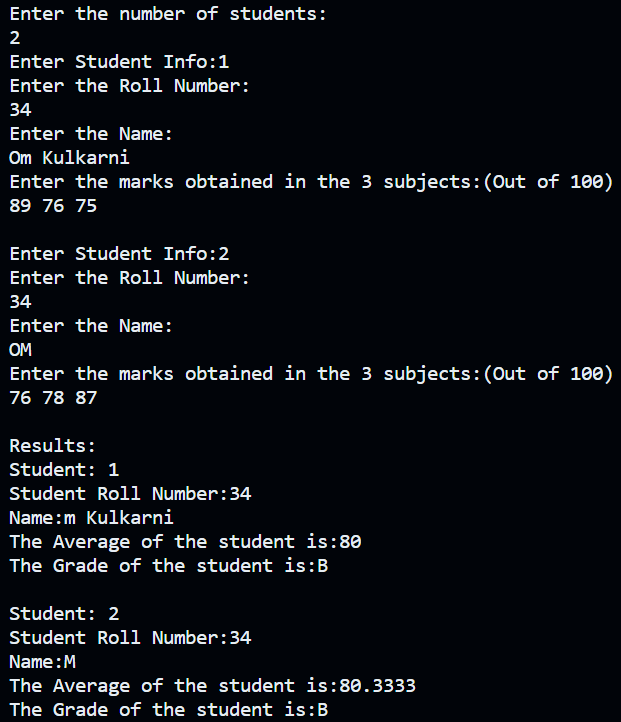
       cout<<endl;

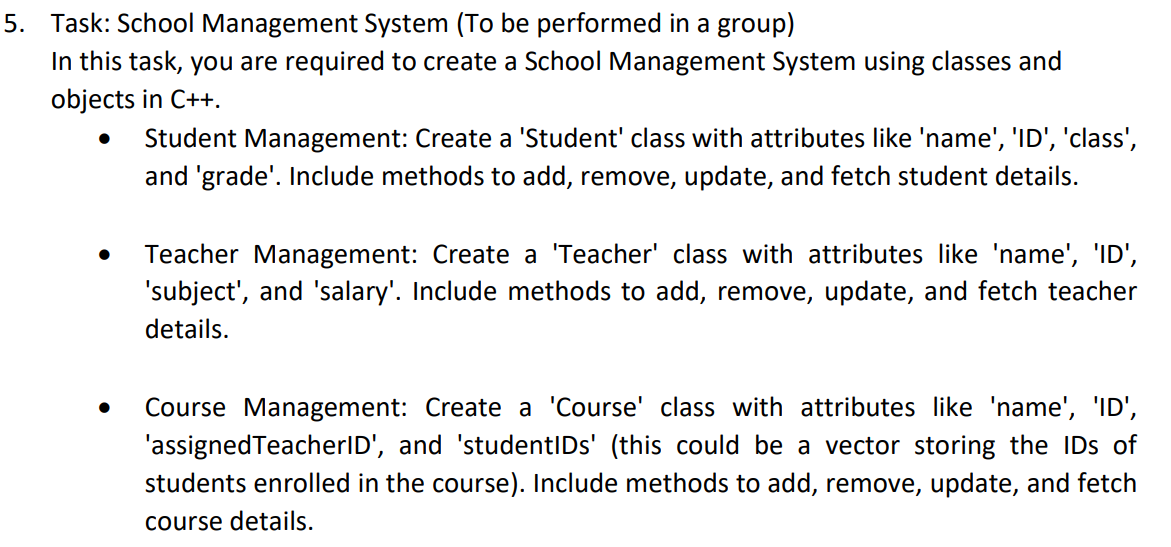
       }

       return 0;

}

OUTPUT:





#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Student

{

public:

    string name, ID, sClass;

    char grade;

    void addStudent();

    void removeStudent();

    void updateStudent();

    void fetchStudent();

};

class Teacher

{

public:

    string name, ID, subject;

    double salary;

    void addTeacher();

    void removeTeacher();

    void updateTeacher();

    void fetchTeacher();

};

class Course

{

public:

    string name, ID;

    string assignedTeacherID;

    vector<string> studentIDs;

    void addCourse();

    void removeCourse();

    void updateCourse();

    void fetchCourse();

};

vector<Student> students;

vector<Teacher> teachers;

vector<Course> courses;

void Student::addStudent()

{

    Student student;

    cout << "Enter the name of the student: ";

    getline(cin, student.name);

    cout << "Enter the ID of the student: ";

    getline(cin, student.ID);

    cout << "Enter the class of the student: ";

    getline(cin, student.sClass);

    cout << "Enter the current grade of the student: ";

    cin >> student.grade;

    cin.ignore();

    students.push\_back(student);

    cout << "Student was successfully added!" << endl;

}

void Student::fetchStudent()

{

    string searchID;

    cout << "Enter the ID of the student: ";

    cin >> searchID;

    for (const auto &student : students)

    {

        if (student.ID == searchID)

        {

            cout << "Name: " << student.name << endl;

            cout << "ID: " << student.ID << endl;

            cout << "Class: " << student.sClass << endl;

            cout << "Grade: " << student.grade << endl;

            return;

        }

    }

    cout << "Student not found." << endl;

}

void Teacher::addTeacher()

{

    Teacher teacher;

    cout << "Enter the name of the teacher: ";

    getline(cin, teacher.name);

    cout << "Enter the ID of the teacher: ";

    getline(cin, teacher.ID);

    cout << "Enter the subject taught by the teacher: ";

    getline(cin, teacher.subject);

    cout << "Enter the salary of the teacher: ";

    cin >> teacher.salary;

    cin.ignore();

    teachers.push\_back(teacher);

    cout << "Teacher was successfully added!" << endl;

}

void Teacher::fetchTeacher()

{

    string searchID;

    cout << "Enter the ID of the teacher: ";

    cin >> searchID;

    for (const auto &teacher : teachers)

    {

        if (teacher.ID == searchID)

        {

            cout << "Name: " << teacher.name << endl;

            cout << "ID: " << teacher.ID << endl;

            cout << "Subject: " << teacher.subject << endl;

            cout << "Salary: " << teacher.salary << endl;

            return;

        }

    }

    cout << "Teacher not found." << endl;

}

void Course::addCourse()

{

    Course course;

    cout << "Enter the name of the course: ";

    getline(cin, course.name);

    cout << "Enter the ID of the course: ";

    getline(cin, course.ID);

    cout << "Enter the ID of the assigned teacher: ";

    getline(cin, course.assignedTeacherID);

    cout << "Enter the number of students to enroll in the course: ";

    int numStudents;

    cin >> numStudents;

    cin.ignore();

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

    {

        string studentID;

        cout << "Enter the ID of student " << i + 1 << ": ";

        getline(cin, studentID);

        course.studentIDs.push\_back(studentID);

    }

    courses.push\_back(course);

    cout << "Course was successfully added!" << endl;

}

void Course::fetchCourse()

{

    string searchID;

    cout << "Enter the ID of the course: ";

    cin >> searchID;

    for (const auto &course : courses)

    {

        if (course.ID == searchID)

        {

            cout << "Name: " << course.name << endl;

            cout << "ID: " << course.ID << endl;

            cout << "Assigned Teacher ID: " << course.assignedTeacherID << endl;

            cout << "Student IDs enrolled: ";

            for (const auto &studentID : course.studentIDs)

            {

                cout << studentID << " ";

            }

            cout << endl;

            return;

        }

    }

    cout << "Course not found." << endl;

}

// ... Existing code ...

void Student::removeStudent()

{

    string removeID;

    cout << "Enter the ID of the student to remove: ";

    cin >> removeID;

    auto it = find\_if(students.begin(), students.end(), [removeID](const Student &student) {

        return student.ID == removeID;

    });

    if (it != students.end())

    {

        students.erase(it);

        cout << "Student removed successfully!" << endl;

    }

    else

    {

        cout << "Student not found." << endl;

    }

}

void Student::updateStudent()

{

    string updateID;

    cout << "Enter the ID of the student to update: ";

    cin >> updateID;

    auto it = find\_if(students.begin(), students.end(), [updateID](const Student &student) {

        return student.ID == updateID;

    });

    if (it != students.end())

    {

        // Updating student attributes

        cout << "Enter the updated name of the student: ";

        cin.ignore();

        getline(cin, it->name);

        cout << "Enter the updated class of the student: ";

        getline(cin, it->sClass);

        cout << "Enter the updated current grade of the student: ";

        cin >> it->grade;

        cout << "Student updated successfully!" << endl;

    }

    else

    {

        cout << "Student not found." << endl;

    }

}

void Teacher::removeTeacher()

{

    string removeID;

    cout << "Enter the ID of the teacher to remove: ";

    cin >> removeID;

    auto it = find\_if(teachers.begin(), teachers.end(), [removeID](const Teacher &teacher) {

        return teacher.ID == removeID;

    });

    if (it != teachers.end())

    {

        teachers.erase(it);

        cout << "Teacher removed successfully!" << endl;

    }

    else

    {

        cout << "Teacher not found." << endl;

    }

}

void Teacher::updateTeacher()

{

    string updateID;

    cout << "Enter the ID of the teacher to update: ";

    cin >> updateID;

    auto it = find\_if(teachers.begin(), teachers.end(), [updateID](const Teacher &teacher) {

        return teacher.ID == updateID;

    });

    if (it != teachers.end())

    {

        cout << "Enter the updated name of the teacher: ";

        cin.ignore();

        getline(cin, it->name);

        cout << "Enter the updated subject taught by the teacher: ";

        getline(cin, it->subject);

        cout << "Enter the updated salary of the teacher: ";

        cin >> it->salary;

        cout << "Teacher updated successfully!" << endl;

    }

    else

    {

        cout << "Teacher not found." << endl;

    }

}

void Course::removeCourse()

{

    string removeID;

    cout << "Enter the ID of the course to remove: ";

    cin >> removeID;

    auto it = find\_if(courses.begin(), courses.end(), [removeID](const Course &course) {

        return course.ID == removeID;

    });

    if (it != courses.end())

    {

        courses.erase(it);

        cout << "Course removed successfully!" << endl;

    }

    else

    {

        cout << "Course not found." << endl;

    }

}

void Course::updateCourse()

{

    string updateID;

    cout << "Enter the ID of the course to update: ";

    cin >> updateID;

    auto it = find\_if(courses.begin(), courses.end(), [updateID](const Course &course) {

        return course.ID == updateID;

    });

    if (it != courses.end())

    {

        cout << "Enter the updated name of the course: ";

        cin.ignore();

        getline(cin, it->name);

        cout << "Enter the updated ID of the assigned teacher: ";

        getline(cin, it->assignedTeacherID);

        cout << "Enter the number of students to enroll in the course: ";

        int numStudents;

        cin >> numStudents;

        cin.ignore();

        it->studentIDs.clear();

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

        {

            string studentID;

            cout << "Enter the ID of student " << i + 1 << ": ";

            getline(cin, studentID);

            it->studentIDs.push\_back(studentID);

        }

        cout << "Course updated successfully!" << endl;

    }

    else

    {

        cout << "Course not found." << endl;

    }

}

int main()

{

    int choice;

    do

    {

        cout << "\nSchool Management System" << endl;

        cout << "1. Student Management" << endl;

        cout << "2. Teacher Management" << endl;

        cout << "3. Course Management" << endl;

        cout << "4. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        cin.ignore();

        switch (choice)

        {

        case 1:

        {

            Student student;

            int studentChoice;

            do

            {

                cout << "\nStudent Management" << endl;

                cout << "1. Add Student" << endl;

                cout << "2. Fetch Student" << endl;

                cout << "3. Update Student" << endl;

                cout << "4. Remove Student" << endl;

                cout << "5. Back" << endl;

                cout << "Enter your choice: ";

                cin >> studentChoice;

                cin.ignore();

                switch (studentChoice)

                {

                case 1:

                    student.addStudent();

                    break;

                case 2:

                    student.fetchStudent();

                    break;

                case 3:

                    student.updateStudent();

                    break;

                case 4:

                    student.removeStudent();

                    break;

                case 5:

                    break;

                default:

                    cout << "Invalid choice." << endl;

                }

            } while (studentChoice != 5);

            break;

        }

        case 2:

        {

            Teacher teacher;

            int teacherChoice;

            do

            {

                cout << "\nTeacher Management" << endl;

                cout << "1. Add Teacher" << endl;

                cout << "2. Fetch Teacher" << endl;

                cout << "3. Update Teacher" << endl;

                cout << "4. Remove Teacher" << endl;

                cout << "5. Back" << endl;

                cout << "Enter your choice: ";

                cin >> teacherChoice;

                cin.ignore();

                switch (teacherChoice)

                {

                case 1:

                    teacher.addTeacher();

                    break;

                case 2:

                    teacher.fetchTeacher();

                    break;

                case 3:

                    teacher.updateTeacher();

                    break;

                case 4:

                    teacher.removeTeacher();

                    break;

                case 5:

                    break;

                default:

                    cout << "Invalid choice." << endl;

                }

            } while (teacherChoice != 5);

            break;

        }

        case 3:

        {

            Course course;

            int courseChoice;

            do

            {

                cout << "\nCourse Management" << endl;

                cout << "1. Add Course" << endl;

                cout << "2. Fetch Course" << endl;

                cout << "3. Update Course" << endl;

                cout << "4. Remove Course" << endl;

                cout << "5. Back" << endl;

                cout << "Enter your choice: ";

                cin >> courseChoice;

                cin.ignore();

                switch (courseChoice)

                {

                case 1:

                    course.addCourse();

                    break;

                case 2:

                    course.fetchCourse();

                    break;

                case 3:

                    course.updateCourse();

                    break;

                case 4:

                    course.removeCourse();

                    break;

                case 5:

                    break;

                default:

                    cout << "Invalid choice." << endl;

                }

            } while (courseChoice != 5);

            break;

        }

        case 4:

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

            break;

        default:

            cout << "Invalid choice." << endl;

        }

    } while (choice != 4);

    return 0;

}

