**Lab 13 - Write a C++ program to implement friend functions based on your domain**

#include <iostream>

#include <string>

using namespace std;

class Student

{

private:

string name;

int rollNo;

float marks[3];

float average;

string grade;

public:

Student(string name = "Unknown", int rollNo = 0, float mark1 = 0.0, float mark2 = 0.0, float mark3 = 0.0)

{

this->name = name;

this->rollNo = rollNo;

this->marks[0] = mark1;

this->marks[1] = mark2;

this->marks[2] = mark3;

this->average = 0.0;

this->grade = "F";

}

Student(const Student &student)

{

name = student.name;

rollNo = student.rollNo;

for (int i = 0; i < 3; i++)

{

marks[i] = student.marks[i];

}

average = student.average;

grade = student.grade;

}

string getName() const

{

return name;

}

float\* getMarks()

{

return marks;

}

void getDetails()

{

cin.ignore();

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter roll number: ";

cin >> rollNo;

cout << "Enter marks for 3 subjects:\n";

for (int i = 0; i < 3; i++)

{

cout << "Subject " << i + 1 << ": ";

cin >> marks[i];

}

}

void calculateAverage()

{

float total = 0;

for (int i = 0; i < 3; i++)

{

total += marks[i];

}

average = total / 3;

}

void calculateGrade()

{

if (average >= 90)

{

grade = "A+";

}

else if (average >= 80)

{

grade = "A";

}

else if (average >= 75)

{

grade = "B+";

}

else if (average >= 70)

{

grade = "B";

}

else if (average >= 60)

{

grade = "C";

}

else

{

grade = "F";

}

}

void displayDetails() const

{

cout << "\nStudent Name: " << name << endl;

cout << "Roll Number: " << rollNo << endl;

cout << "Marks in 3 subjects:";

for (int i = 0; i < 3; i++)

{

cout << " " << marks[i];

}

cout << endl;

cout << "Average Marks: " << average << endl;

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

}

friend void compareMarks(const Student &s1, const Student &s2);

friend void checkEligibility(const Student &s);

friend void applyAttendanceBonus(Student &s, float attendance);

friend void applyLowAttendancePenalty(Student &s, float attendance);

};

void compareMarks(const Student &s1, const Student &s2)

{

cout << "\nComparing students " << s1.getName() << " and " << s2.getName() << ":\n";

if (s1.average > s2.average)

{

cout << s1.getName() << " has higher average marks: " << s1.average << endl;

}

else if (s2.average > s1.average)

{

cout << s2.getName() << " has higher average marks: " << s2.average << endl;

}

else

{

cout << "Both have the same average marks: " << s1.average << endl;

}

}

void checkEligibility(const Student &s)

{

cout << "\nChecking eligibility of " << s.getName() << " for award:\n";

if (s.average >= 85)

{

cout << s.getName() << " is eligible for the award." << endl;

}

else

{

cout << s.getName() << " is not eligible for the award." << endl;

}

}

void applyAttendanceBonus(Student &s, float attendance)

{

if (attendance > 90)

{

cout << s.getName() << " gets a 5% bonus on marks for excellent attendance!" << endl;

for (int i = 0; i < 3; i++)

{

s.marks[i] += s.marks[i] \* 0.05;

}

}

else

{

cout << s.getName() << " does not qualify for the attendance bonus." << endl;

}

}

void applyLowAttendancePenalty(Student &s, float attendance)

{

if (attendance < 75)

{

cout << s.getName() << " receives a 10% penalty on total marks for low attendance." << endl;

for (int i = 0; i < 3; i++)

{

s.marks[i] -= s.marks[i] \* 0.10;

}

}

}

int main()

{

Student defaultStudent;

defaultStudent.calculateAverage();

defaultStudent.calculateGrade();

defaultStudent.displayDetails();

Student paramStudent("Alice", 101, 85.5, 92.0, 78.5);

paramStudent.calculateAverage();

paramStudent.calculateGrade();

paramStudent.displayDetails();

Student userStudent;

userStudent.getDetails();

userStudent.calculateAverage();

userStudent.calculateGrade();

userStudent.displayDetails();

Student copyStudent(userStudent);

cout << "\nDetails of copied student:\n";

copyStudent.displayDetails();

compareMarks(paramStudent, userStudent);

checkEligibility(paramStudent);

float attendance;

cout << "\nEnter attendance percentage for " << paramStudent.getName() << ": ";

cin >> attendance;

applyAttendanceBonus(paramStudent, attendance);

applyLowAttendancePenalty(paramStudent, attendance);

paramStudent.calculateAverage();

paramStudent.calculateGrade();

paramStudent.displayDetails();

return 0;

}