#include <iostream>

#include <string>

#include <vector>

class College {

private:

std::string name;

std::string location;

public:

College(std::string name, std::string location) : name(name), location(location) {}

std::string getName() const {

return name;

}

std::string getLocation() const {

return location;

}

};

class DateOfBirth {

private:

int day, month, year;

public:

DateOfBirth(int d, int m, int y) : day(d), month(m), year(y) {}

// Function to calculate the age of the student

int calculateAge(int currentYear, int currentMonth, int currentDay) const {

int age = currentYear - year;

if (currentMonth < month || (currentMonth == month && currentDay < day)) {

age--;

}

return age;

}

};

class Student {

public:

std::string name;

std::string regNumber;

private:

std::string course;

std::vector<int> marks;

double GPA;

double CGPA;

int semester;

College college;

DateOfBirth dob;

public:

// Constructor to initialize student attributes

Student(std::string name, std::string regNumber, std::string course, College college, DateOfBirth dob)

: name(name), regNumber(regNumber), course(course), college(college), dob(dob), GPA(4.5), CGPA(4.5), semester(1) {}

// Getter for course

std::string getCourse() const {

return course;

}

// Setter for course

void setCourse(std::string c) {

course = c;

}

// Getter for GPA

double getGPA() const {

return GPA;

}

// Function to update GPA based on the marks

void calculateGPA() {

if (marks.empty()) {

GPA = 4.5; // Default GPA if no marks are present

return;

}

int totalMarks = 0;

for (int mark : marks) {

totalMarks += mark;

}

GPA = static\_cast<double>(totalMarks) / marks.size();

}

// Function to add marks for the current semester

void addMarks(const std::vector<int>& semesterMarks) {

marks = semesterMarks;

calculateGPA();

updateCGPA();

}

// Getter for CGPA

double getCGPA() const {

return CGPA;

}

// Function to update CGPA after each semester

void updateCGPA() {

if (semester == 1) {

CGPA = GPA; // For the first semester, CGPA is equal to GPA

} else {

CGPA = (CGPA \* (semester - 1) + GPA) / semester; // Average over semesters

}

}

// Function to track and update the current semester

void updateSemester() {

semester++;

}

// Function to get the student's age using the DateOfBirth object

int getAge(int currentYear, int currentMonth, int currentDay) const {

return dob.calculateAge(currentYear, currentMonth, currentDay);

}

};

int main() {

// Sample usage

College college("Tech University", "New York");

DateOfBirth dob(15, 8, 2000);

Student student("John Doe", "TU12345", "Computer Science", college, dob);

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

std::cout << "Registration Number: " << student.regNumber << std::endl;

std::cout << "College: " << college.getName() << ", " << college.getLocation() << std::endl;

// Input marks and calculate GPA

std::vector<int> marks = {85, 90, 88, 92};

student.addMarks(marks);

std::cout << "GPA: " << student.getGPA() << std::endl;

std::cout << "CGPA: " << student.getCGPA() << std::endl;

// Update semester and add more marks

student.updateSemester();

std::vector<int> newMarks = {80, 85, 90, 95};

student.addMarks(newMarks);

std::cout << "Updated GPA: " << student.getGPA() << std::endl;

std::cout << "Updated CGPA: " << student.getCGPA() << std::endl;

// Calculate and display age

int currentYear = 2024, currentMonth = 10, currentDay = 9;

std::cout << "Age: " << student.getAge(currentYear, currentMonth, currentDay) << " years" << std::endl;

return 0;

}