ASSIGNMENT 04
PROGRAMMING FUNDAMENTALS

INSTRUCTIONS:

1) Plagiarism is strictly forbidden. Each individual must complete the work independently as it is part of the learning process.

2) The naming format must be strictly followed as given in the instruction note:

23P-0603_Aimalkhan_Q1.c

23P-0603_Aimalkhan_Q1.docs

3) Anyone who fails to follow the naming format will receive zero marks.

4) Everything covered in class so far is included in the assignment.

5) Comment your code properly.

6) Upload word file of screenshots.

7) Anyone failed to follow these instructions will get a straight zero.

8) Don't use AI for this assignment try to do it on your own it will be helpful in your sessional exam.

9) Understanding is the part of assignment.

1. Objective

The purpose of this assignment is to design and implement a complete University Academic & Attendance Management System using only multidimensional arrays in C. This assignment will assess your concepts of arrays, data manipulation, searching, sorting, functions, input validation, and logical thinking.

2. Problem Description

You are required to develop a menu driven academic management system that handles:

1) Student records
2) Course records

3) Marks, attendance, GPA calculation

4) Academic analysis & reporting

5)Searching and sorting

6)Performance analytics

The system should handle up to:

1) 50 students

2) 10 courses

This assignment simulates a real university environment where students enroll in multiple courses, and the system must track their academic progress, attendance, GPA, and performance statistics.

3. Data Storage Requirements (Must Use Multidimensional Arrays)

A. Course Data (Max 10 courses)

Each course has:

1. Course ID
2. Course Name
3. Maximum Marks
4. Credit Hours

Suggested structure:
courses[10][4] Store values as strings (convert numbers using stoi() when needed).

B. Student Data (Max 50 students)

Each student has:

1. Student ID
2. Student Name
3. Their enrolled course IDs (multiple per student)

C. Marks & Attendance Storage

marks[50][10] Marks per student per course

attendance[50][10] Attendance percentage

D. GPA Calculation Formula

```
GPA = (Marks / MaxMarks) * 4.0
```

4. System Functionalities

You must implement all the following features using functions:

A. Add Course

Input: ID, Name, Max Marks, Credit Hours

Validate ID (must be unique)

B. Add Student

Input: Student ID, Name

Choose courses for enrollment

Initialize marks & attendance to zero

C. Update Marks

Input: Student ID, Course ID, Marks

Validate marks (cannot exceed max marks)

D. Update Attendance

Input: Student ID, Course ID, Percentage

Validate (0–100 allowed only)

E. Calculate GPA

For each student:

GPA per course

Cumulative GPA (considering credit hours)

F. Display Student Report

Input: Student ID
Must display:

All enrolled courses

Marks

Attendance

GPA (course-wise & cumulative)

Status for each course:

Pass: Marks ≥ 50 AND Attendance ≥ 75%

Fail otherwise

G. Display Course Report

Input: Course ID
Display:

Students enrolled

Their marks, attendance, GPA

Highest marks

Lowest marks

Average marks

Average attendance

H. Topper by Cumulative GPA

Identify:

- Student(s) with highest cumulative GPA

I. Attendance Warning List

Display:

- Students with attendance < 75% in any course

J. Course Performance Analysis

For each course:

- Pass percentage
- Professor rating indicator (optional)
- Course with:
  - Highest average marks
  - Lowest average marks

K. **Search Functionality**

- Search student by ID or name
- Search course by ID or name

L. Sorting

Sort using your own algorithm (Bubble, Selection, or Insertion):

- Sort students by GPA
- Sort students by total marks

Sort courses by:

Average marks

Enrollment size

5. Constraints

1) Multidimensional arrays only

2) No structs

3) No vectors

4) No classes

5) No file handling

6) No pointers to structures

7) Only procedural programming

8) Functions are mandatory

9) At least 10 students and 5 courses in sample demonstration

10) Input validation required everywhere

7. Output Requirements

Reports must be neat, readable, formatted using proper spacing:

Use tables

Use headings

Align columns properly

Display summary statistics