<u>Assignment-1</u>

Designing a Modular Student Performance Analyzer in Python
(25 marks + 5 bonus marks)

You are required to build a console-based Student Performance Analyzer that processes student data, computes grades, stores records, and generates simple reports.

The system must:

1. Allow the instructor to add, remove, search, and update student records.
2. Compute a final grade based on three exam scores using a formula:
   final_score = (0.4 * midterm) + (0.5 * final_exam) + (0.1 * assignment_score)
3. Convert the numerical score into a letter grade using conditions:
```
A: 85-100
B: 70-84
C: 55-69
D: 40-54
F: <40
```
4. Generate summary statistics, including:
   a. Highest & lowest scores
   b. Class average
   c. Grade distribution (count per letter grade)
5. Store all records in a CSV file and load them on startup.

Task 1 — Basic Input & Data Storage (5 Marks)

- Create a Python file named student_analyzer.py.
- Initialize data storage using a list of dictionaries, where each dictionary represents a student:

```
{
  "id": 101,
  "name": "Ali",
  "midterm": 78,
  "final": 85,
  "assignment": 90,
  "final_score": 0,
  "grade": ""
}
```

Task 2 — Functions for Core Operations (5 Marks)

Write functions for:

- add_student()
- remove_student()
- update_student()
- search_student()
- calculate_final_score()
- assign_letter_grade()
- show_all_students()

Each function must use loops, conditionals, and list/dictionary operations.

Task 3 — File Handling (4 Marks)

- On startup, load student records from a CSV file (students.csv).
- On exit, save updated records back to the file.
- Handle missing file errors using try/except.

Task 4 — Object-Oriented Programming (5 Marks)

Define a class:

```
class Student:
    def __init__(self, id, name, midterm, final, assignment):
        # initialize attributes here

    def compute_final(self):
        # compute score using formula

    def compute_grade(self):
        # determine letter grade

    def to_dict(self):
        # return dictionary representation
```

Integrate this class into your system.

Task 5 — Reporting & Statistics (6 Marks)

Create a function generate_report() that calculates:

- Total number of students
- Highest and lowest final score
- Average final score
- Grade frequency distribution
- A sorted list by final score (descending)
- Display results neatly in the console.

Task 6 — Bonus Optional (5 Marks)

Add:

- Grade Visualization

Required Documentation

1. student_analyzer.py
2. students.csv (sample data)
3. PDF file containing:
    a. Flowchart OR pseudocode
    b. Screenshots of program output
    c. Explanation of challenges and learning outcomes

Learning Outcomes

After completing this assignment, students will be able to:

- Build non-trivial Python programs
- Use functions and classes effectively
- Manipulate data structures (list, dict, file I/O)
- Apply reasoning, logic, and structured programming
- Develop real-world console applications

Deadline: We will grade the assignment in the class before terminal exams.

****