Low-Level Design (LLD) for Exam Score Analyzer

Difficulty Level: Easy | **Total Marks:** 10
Standards Followed: 4 Functions | 3 Visible Test Cases | 2 Hidden Test Cases

>> Concepts Tested

-- NumPy array creation and usage
-- Average, max, min calculations
-- Filtering using NumPy conditions
-- Classification using loops and thresholds

>> Problem Statement

Create a system to analyze exam scores. The program will accept student marks, calculate the average, find the highest and lowest scores, filter passing scores (>=40), and assign grades. All operations should be encapsulated in a class using NumPy arrays, with methods returning values for processing.

>> Operations

1. **Load Scores**
   Function Prototype: def load_scores(self, score_list: list):
   Example Input: load_scores([55, 89, 72, 100])
   Expected Return: np.array([55, 89, 72, 100])
2. **Compute Summary**
   Function Prototype: def compute_summary(self):
   Expected Return: "Average: 79.0, Max: 100, Min: 55"
3. **Filter Passing Scores**
   Function Prototype: def get_passing_scores(self):
   Expected Return: np.array([55, 89, 72, 100])
4. **Assign Grades**
   Function Prototype: def assign_grades(self):
   Expected Return: ['D', 'B', 'C', 'A']

>> Implementation Code

```
import numpy as np


class ExamScoreAnalyzer:

    def __init__(self):

        """Initializes an empty NumPy array to store exam scores."""

        self.scores = np.array([])
```

```python
    def load_scores(self, score_list):
        """
        Loads a list of exam scores into the analyzer and returns the array.
        Example:
        Input: [55, 89, 72, 100]
        Return: np.array([55, 89, 72, 100])
        """
        # TODO: Implement this method
        pass

    def compute_summary(self):
        """
        Calculates and returns the average, maximum, and minimum score as a string.
        Return Format: "Average: X.X, Max: Y, Min: Z"
        """
        # TODO: Implement this method
        pass

    def get_passing_scores(self):
        """
        Returns scores that are greater than or equal to 40.
        Example:
        Return: np.array([55, 89, 72, 100])
        """
        # TODO: Implement this method
        pass

    def assign_grades(self):
        """
        Returns grades based on scores:
        ≥90 → 'A', 75-89 → 'B', 60-74 → 'C', 40-59 → 'D', <40 → 'F'
        Example:
        Return: ['D', 'B', 'C', 'A']
        """
        # TODO: Implement this method
        pass

# Driver Code
if __name__ == "__main__":
    analyzer = ExamScoreAnalyzer()
    q = int(input())  # Number of operations
    for _ in range(q):
        command = input().split()
        if command[0] == "load":
            scores = analyzer.load_scores(list(map(int, command[1:])))
            print(scores)
        elif command[0] == "summary":
            summary = analyzer.compute_summary()
            print(summary)
        elif command[0] == "pass":
            passing = analyzer.get_passing_scores()
            print(passing)
        elif command[0] == "grades":
            grades = analyzer.assign_grades()
            print(grades)
        else:
            print("Invalid command.")
```

>> Test Cases & Marks Allocation

Test Case ID	**Test Case Description**	**Associated Function(s)**	**Marks**
TC1	Loading exam scores into the array	load_scores()	-- 2 Marks
TC2	Computing average, max, and min	compute_summary()	-- 2 Marks
TC3	Filtering passing scores	get_passing_scores()	-- 2 Marks
HTC1	Assigning letter grades using thresholds	assign_grades()	-- 2 Marks
HTC2	Handling failing scores (<40)	assign_grades()	-- 2 Marks
TOTAL	All test cases passed	–	☐ 10 Marks

>> Visible Test Cases (3)

Test Case 1: Load Scores
Input:

```
1
load 55 89 72 100
```

Output: [55 89 72 100]

Test Case 2: Compute Summary
Input:

```
2
load 55 89 72 100
summary
```

Output: Average: 79.0, Max: 100, Min: 55

Test Case 3: Get Passing Scores
Input:

```
2
load 55 89 72 100
pass
```

Output: [55 89 72 100]

>> Hidden Test Cases (2)

HTC1: Assign Grades
Input:

```
2
load 55 89 72 100
```

```
grades
```

Output: ['D', 'B', 'C', 'A']

HTC2: Handle Failing Scores
Input:

```
2
load 25 40 75
grades
```

Output: ['F', 'D', 'B']