SE LAB TASK 2

24MCMT20 PARNAB DUTTA

MODULE SPECIFICATION

Application: Student Result Processing System

The application is divided into multiple independent modules to improve readability, reusability, maintainability and ease of testing. Each module performs a specific task and interacts with other modules using well-defined interfaces.

1. Validation Module

Module Name

Validation Module

Functions

```
validId()
```

```
validName()
```

```
validMark()
```

Input

Student ID (string)

Student Name (string)

Marks (integer)

Pre-Condition

Input values must be read from the file before validation.

Logic

`validId()` checks whether the ID contains only alphanumeric characters.

`validName()` checks whether the name contains only alphabetic characters.

`validMark()` verifies that marks are within the range 0 to 100.

Output

Returns 1 if input is valid.

Returns `0` if input is invalid.

Purpose

This module ensures data correctness and prevents invalid student records from being processed.

2. CGPA Module

Module Name

CGPA Module

Function

```
calculateCgpa()
```

Input

Grade (string)

Pre-Condition

Grade must be assigned before CGPA calculation.

Logic

The function maps grades such as O, A+, A, B+, B, C, D and F to their respective CGPA values.

Invalid grades result in CGPA value 0.

Output

Returns CGPA as a floating-point value.

Purpose

This module separates CGPA computation logic, making it reusable across different applications.

3. Grade Module

Module Name

Grade Module

Functions

```
assignGrade()
```

`gradeIndex()`

Input

Student percentage

Student grade

Pre-Condition

Total marks and percentage must be calculated before calling this module.

Logic

`assignGrade()` assigns grade based on percentage slabs and then calls the CGPA module to calculate CGPA.

`gradeIndex()` converts grade into an index value used for grade distribution analysis.

Output

Assigned grade

Calculated CGPA

Grade index value

Purpose

This module handles grading logic and grade-related statistics.

4. Student Module

Module Name

Student Module

Functions

`readStudents()`

`printResults()`

Input

Student records from `students.txt`

Student structure array

Pre-Condition

Input file must exist and be readable.

Logic

`readStudents()` reads student data, validates inputs using the validation module, calculates totals, percentages, grades and CGPA.

`printResults()` displays student details, class average, highest and lowest percentage, and grade distribution.

Output

- Populated student records
- Printed result table and statistics

Purpose

This module manages student data processing and output generation.

5. Main Module

Module Name

Main Module

Function

- `main()`

Input

- None (controls program execution)

Pre-Condition

All required modules must be correctly linked.

Logic

- Calls `readStudents()` to load and process student data.
- Calls `printResults()` to display results.

Output

- Complete execution of the student result processing system.

Purpose

Acts as the entry point and coordinates interaction between all modules.

Conclusion

The modular design ensures:

- Low coupling and high cohesion
- Easy testing and maintenance
- Reusability of individual modules
- Compliance with Software Engineering best practices