Report

Name :J Krishna Charan
25MCMI26

Detailed Module Functionality & Specifications

Each module has a unique name, is separately compiled, and maintains high cohesion as required by the project guidelines.

1.1 Preprocessing Module (Preprocessing.c)

Module Name: Data Validation and Sanitization.

- **Input**: Character arrays (ID, Name) and the current student array.
- **Pre-condition**: Raw strings must be successfully read from the input file before validation.
- **Output**: Boolean (Integer 1/0) indicating if the record is valid or a duplicate.

Function Details & Logic:

- **ID_Check(id)**: Iterates through the ID string to ensure every character is alphanumeric (ASCII 48–57, 65–90, or 97–122).
- **Name_Check(name)**: Validates that names contain only alphabetic characters.
- **Duplicate_Check(list, count, id)**: Performs a linear search through existing records to prevent redundant data entry.

Pseudocode:

```
FUNCTION ID_Check(id):
   FOR each character in id string:
      IF character IS NOT (0-9 OR A-Z OR a-z):
         RETURN 0 (Invalid)
   RETURN 1 (Valid)

FUNCTION Name_Check(name):
   FOR each character in name string:
      IF character IS NOT (A-Z OR a-z):
         RETURN 0 (Invalid)
   RETURN 1 (Valid)
```

```
FUNCTION Duplicate_Check(students, count, current_id):
   FOR i FROM 0 TO count - 1:
      IF current_id matches students[i].id:
         RETURN 1 (Duplicate Found)
   RETURN 0 (Unique)
```

1.2 Grading Engine Module (Grading.c)

Module Name: Grading.

- **Input**: A pointer to a student structure containing marks.
- **Pre-condition**: Marks must be within the valid range (Minor: 0–40, Major: 0–60).
- **Output**: Updated student structure with individual subject grades and a final weighted CGPA.

Academic Logic Details:

The system employs a credit-weighted system where subjects carry specific weights: **{3, 4, 3, 3, 2}**.

- **Grading Scale**:

  90+: O (10 GP)

  85–89: A+ (9 GP)

  75–84: A (8 GP)

  65–74: B+ (7 GP)

  60–64: B (6 GP)

  55–59: C (5 GP)

  50–54: D (4 GP)

  Below 50: F (0 GP

Pseudocode:

```
FUNCTION assignGrade(student_ptr):
   SET credits = {3, 4, 3, 3, 2}
   SET total_weighted_gp = 0
   SET fail_flag = 0

   FOR each subject (0 to 4):
      total_score = student_ptr->marks[subject].minor +
student_ptr>marks[subject].major

      IF total_score >= 90: grade = "O", gp = 10
      ELSE IF total_score >= 85: grade = "A+", gp = 9
      ...
      ELSE IF total_score < 50: grade = "F", gp = 0, fail_flag = 1

      total_weighted_gp += (gp * credits[subject])

   student_ptr->cgpa = total_weighted_gp / SUM(credits)

   IF fail_flag IS 1:
      student_ptr->final_grade = "F"
   ELSE:
      Assign final_grade based on CGPA threshold
```

1.3 Display & Analytics Module (Display.c)

Module Name: Reporting and Statistics.

- **Input**: Array of processed student records and the total count.
- **Pre-condition**: The student array must contain at least one valid record.
- **Output**: A formatted console table and class-wide performance metrics.

Function Details & Logic:

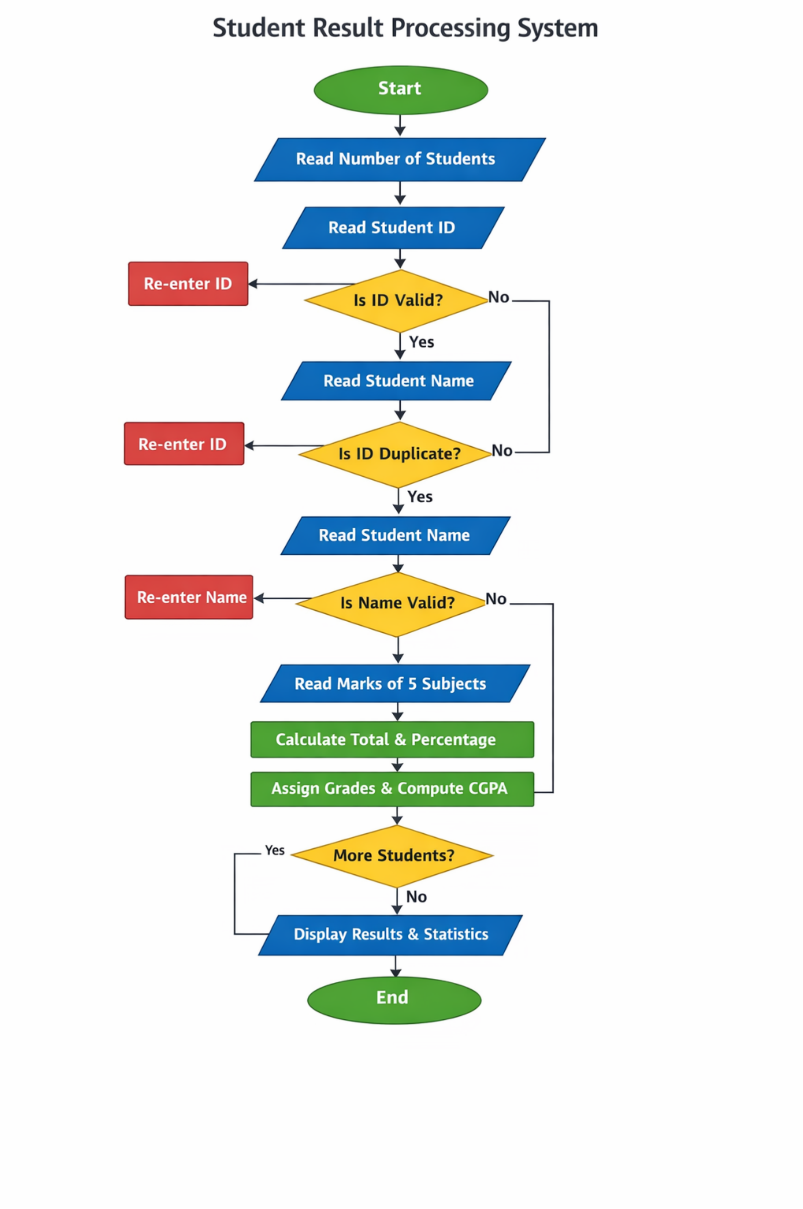
- **Display(students, count)**: Generates a summary including class average, the highest percentage, and the lowest percentage achieved.

Pseudocode:

```
FUNCTION Display(students, count):
   PRINT Table Headers (ID, Name, Total, Percentage, Grade, CGPA)
   FOR i FROM 0 TO count - 1:
      PRINT student[i] record
      sum_percentages += students[i].percentage
      IF students[i].percentage > highest: highest = students[i].percentage
      IF students[i].percentage < lowest: lowest = students[i].percentage
      INCREMENT Grade_Counter for student[i].grade

   PRINT Class Statistics:
      Average = sum_percentages / count
      PRINT highest, lowest, and Grade Distribution
```

2. Program Flow



3. Main Execution Flow (main.c)

The main module acts as the orchestrator for the entire application, managing the lifecycle and file interaction:

1. **File Input**: Opens the data file passed as a command-line argument.
2. Processing Loop:
   - Reads student ID and Name.
   - Invokes ID_Check, Name_Check, and Duplicate_Check.
   - Validates mark ranges (0–40 for Minor, 0–60 for Major).
   - Calculates Percentage and calls assignGrade.
3. **Finalization**: Closes the file and triggers the Display module to output the final statistics report.

Git Link for Evaluation:

https://github.com/krishnacharanjamalla/Software-Engineering-Lab-01