DOCUMENTATION

Student Result Processing System Module Specification

Module 1 : MAIN Module

Module Name : Main(main.c)

Input :

File input :student_data.txt

The text file contains Number of students , Student ID ,Name ,Marks of Five Subjects.

Precondition:

- Input file must exist
- File must be readable
- Header file myheader.h must be included

Logic :

Algorithm:

```
IF file cannot be opened
DISPLAY error message
 RETURN 1
  END IF
READ number of students N
CREATE array students[N]
 SET validstudents = 0
FOR n = 0 to N1
   READ student ID, name, and marks from file
    IF validID(buffer.id) is FALSE
      DISPLAY invalid ID message
       CONTINUE
    END IF
   IF validName(buffer.name) is FALSE
     DISPLAY invalid name
       CONTINUE to next student
```

```
      END IF
      FOR each mark
        IF validMarks(mark) is FALSE
          DISPLAY invalid marks message
          SKIP this student
        END IF
      END FOR
      IF uniqueID(buffer.id, students, validstudents) is FALSE
        DISPLAY duplicate ID message
        CONTINUE
      END IF
      ADD buffer to students array
      INCREMENT validstudents
    END FOR
    CLOSE file
  CALL displayverifiedstudent(students,validstudents)
    FOR i = 0 to validstudents1
      students[i].total = marksTotal(students[i].marks)
      students[i].percentage = calculatePercentage(students[i].total)
      IF checkPassSubjects(students[i].marks)
        gradeassignment(students[i].percentage, students[i].grade)
      ELSE
        students[i].grade = "F"
      END IF
      students[i].CGPA = calculateCGPA(students[i].percentage)
    END FOR
    CALL displayReport(students, validstudents)
    CALL percentageStats(students, validstudents)
    CALL gradeStats(students, validstudents)
```

RETURN 0

Output:

Displays invalid students data

Displays validated student data

Displays final Result table

Displays statistics

Module 2 : VALIDATION Module(validation.c)

Module Name : Validation

Function 1: validID()

Input: char id[] Student ID string

Precondition: ID string should not be NULL

Logic:

Algorithm:

FOR each character in id string

IF character is NOT alphanumeric

RETURN 0 (invalid)

END IF

END FOR

RETURN 1 (valid)

Output: Returns 1 if valid (contains only alphanumeric characters), 0 otherwise

Function 2: validName

Input: char name[] Student name string

Precondition: Name string terminate with NULL

Logic:

Algorithm:

FOR each character in name string

IF character is NOT alphabetic

RETURN 0 (invalid)

END IF

END FOR

RETURN 1 (valid)

Output: Returns 1 if valid (contains only alphabetic characters), 0 otherwise

Function 3: validMarks()

Input: int marks Single subject marks

Precondition: None

Logic:

Algorithm:

IF marks < 0 OR marks > MAX_MARKS

RETURN 0 (invalid)

ELSE

RETURN 1 (valid)

END IF

Output: Returns 1 if marks are in range [0, MAX_MARKS], 0 otherwise

Function 4: uniqueID()

Input:

char id[] Student ID

struct Student students[] Array of existing students

int count Number of existing students

Precondition: count >= 0, students array contains valid data

Logic:

Algorithm:

FOR i = 0 to count1

IF id matches students[i].id

RETURN 0 (duplicate found)

END IF

END FOR

RETURN 1 (unique)

Output: Returns 1 if ID is unique, 0 if duplicate found

Module 3 : COMPUTATION Module

Module Name :computation

Input :

Function 1: marksTotal

Input: int marks[] Array of marks for all subjects

Precondition: Array has SUBJECTS elements

Logic:

Algorithm:

SET total = 0

FOR i = 0 to SUBJECTS1

total = total + marks[i]

END FOR

RETURN total

Output: Returns sum of all marks

Function 2: calculatePercentage

Input: int total Total marks obtained

Precondition: total >= 0

Logic:

Algorithm:

percentage = total / SUBJECTS

RETURN percentage

Output: Returns percentage as total marks / number of subjects

Function 3: calculateCGPA

Input: float percentage Student percentage

Precondition: percentage >= 0

Logic:

Algorithm:

CGPA = percentage / 10

RETURN CGPA

Output: Returns CGPA as percentage / 10

Function 4: checkPassSubjects

Input:int marks[] Array of marks for all subjects

Precondition: Array has SUBJECTS elements

Logic:

Algorithm:

FOR i = 0 to SUBJECTS1

    IF marks[i] < 50

        RETURN 0 (failed)

    END IF

END FOR

RETURN 1 (passed all subjects)

Output: Returns 1 if all subjects have marks >= 50, 0 otherwise

Module 4: ASSIGNMENT Module

Module Name :assignment

Function 5: gradeassignment

Input:

float percentage : Student percentage

char grade[] : Grade string

Precondition: percentage >= 0

Logic:

Algorithm

IF percentage >= 90

grade = "O"

ELSE IF percentage >= 85

grade = "A+"

ELSE IF percentage >= 75

grade = "A"

ELSE IF percentage >= 65

grade = "B+"

ELSE IF percentage >= 60

grade = "B"

ELSE IF percentage >= 55

grade = "C"

ELSE IF percentage >= 50

grade = "D"

ELSE

grade = "F"

END IF

Output: Modifies grade array with assigned grade string

Module 5: STATISTICS Module

Module Name :statistics

Function 1: percentageStats

Input:

Array of student records

Number of students

Precodition: count > 0

Logic:

Algorithm:

SET sum = 0

SET highest = students[0].percentage

SET lowest = students[0].percentage

FOR i = 0 to count1

sum = sum + students[i].percentage

IF students[i].percentage > highest

highest = students[i].percentage

END IF

IF students[i].percentage < lowest

10. lowest = students[i].percentage

11. END IF

12. END FOR

13. average = sum / count

14. DISPLAY average, highest, lowest

Output: Displays class average, highest, and lowest percentage

Function 2: gradeStats

Input:

Array of student records

Number of students

Precondition: count > 0

Logic:

Algorithm:

INITIALIZE all grade counter to 0

FOR i = 0 to count1

IF students[i].grade == "O"

INCREMENT countO

ELSE IF students[i].grade == "A+"

INCREMENT countAplus

```
    ELSE IF students[i].grade == "A"
        INCREMENT countA
    ELSE IF students[i].grade == "B+"
        INCREMENT countBplus
    ELSE IF students[i].grade == "B"
        INCREMENT countB
    ELSE IF students[i].grade == "C"
        INCREMENT countC
    ELSE IF students[i].grade == "D"
        INCREMENT countD
    ELSE
        INCREMENT countF
    END IF
  END FOR
```

Output: Displays count of students in each grade category (O, A+, A, B+, B, C, D, F)

Module 6 : DISPLAY Module

Module Name :display

Function 1: displayverifiedlist

Input:

Array of verified student records

Number of valid students

Precondition: validstudents >= 0

Logic:

Algorithm:

```
  FOR i = 0 to validstudents1
     DISPLAY students[i].id
     DISPLAY students[i].name
    FOR j = 0 to SUBJECTS1
```

DISPLAY students[i].marks[j]

END FOR

DISPLAY newline

END FOR

Output: Displays formatted table of verified student data showing ID, name, and marks for all subjects

Function 2: displayReport

Input:

Array of student records with all calculated fields

Number of students

Precondition: count >= 0

Logic:

Algorithm:

FOR i = 0 to count1

DISPLAY students[i].id

DISPLAY students[i].name

FOR j = 0 to SUBJECTS1

DISPLAY students[i].marks[j]

DISPLAY students[i].total

DISPLAY students[i].percentage

DISPLAY students[i].grade

DISPLAY students[i].CGPA

DISPLAY newline

END FOR

Output: Displays formatted table of all student results including calculated values (total, percentage, grade, CGPA)