Average Marks & Grade Calculator for Semester I

1. Research

Problem Identified:

The problem was identified after reviewing the challenges faced by educational institutions in calculating and classifying student grades. The manual system is error-prone and time-consuming, leading to inefficiencies in evaluating students' performance.

The primary problem this project aims to solve is:

- Automating the process of calculating student marks and grades to save time and reduce human error.
- Ensuring students meet the Continuous Internal Evaluation (CIE) criteria before proceeding to grade calculation.

References:

What is Grade?

Simple Grade Calculator

Example in Use

Officially in use Calculator

Importance:

The accuracy and speed of calculating marks for large batches of students are crucial in modern educational systems. Academic professionals need reliable systems to compute averages and categorize grades, which directly impacts students' academic outcomes and institutional efficiency.

Sources:

- Academic institution feedback highlighting the manual burden of grading.
- Research articles on automated grade calculation and student performance analytics.
- Discussions with teachers and faculty members on common grading issues.

2. Analyse

Solution:

The program benefits students and academic institutions by:

- Providing real-time validation of marks and eligibility based on CIE scores.
- Automating the process of calculating student averages and grade classifications, which reduces manual effort.
- Offering transparency in grade assignment, ensuring fairness.

Benefits:

- The program saves time by eliminating manual calculations.
- It reduces human error, ensuring marks and grades are computed with precision.
- Teachers and administrators can rely on the program to validate student eligibility for examinations based on CIE scores.

3. Ideate

How the Program Works (with proper cases):

The program allows batch-wise input for students' marks across five subjects. Key steps include:

- Input validation: The program checks if marks are within the range of 0-100 and ensures that students meet the CIE requirement (75/100).
- Average Calculation: For valid students, the average marks are computed based on subject scores.
- Grade Assignment: Grades (Distinction, First Class, Second Class, Third Class) are assigned based on percentage ranges.

Cases:

- **Valid Input Case**: If all inputs are within valid ranges and CIE is above 75, the average is computed, and the appropriate grade is displayed.
- **Invalid Input Case**: If a student's CIE is less than 75, the program skips that student, indicating they are not eligible.
- **Failing Case**: If marks in all subjects are below the passing criterion (45 marks), the program declares the student as failed.

4. Build

Error Handling Response:

The program includes comprehensive error handling to manage:

- **Invalid Marks Input**: If a mark is outside the range of 0-100, the user is prompted to re-enter the marks.
- **Invalid CIE**: If the CIE is below 75, the program immediately notifies the user and skips that student for further grade computation.

Error handling ensures that the program functions robustly even when unexpected or incorrect data is entered, preventing crashes and providing clear feedback to users.

We have declared and initialized variables:

```
int main() {
    int num; // Number of students
    float math, chem, phy, eng, cs, average, cie;
    float totalAverage = 0; // Total average for the batch
    float totalMath = 0, totalChem = 0, totalPhy = 0, totalEng = 0, totalCs = 0;
    int validStudents = 0;  // Count of eligible students
```

Introduction and input number of students:

```
printf("\nAverage Marks & Grade Calculator for Semester I");

printf("\nNumber of Students in Batch: ");
scanf("%d", &num);
```

Input CIE and check eligibility:

```
for (int i = 0; i < num; i++) {
    printf("\n\nFor Student %d:\n", i + 1);

    // Enter and validate CIE marks
    printf("Enter the CIE for Student %d (0-100): ", i + 1);
    scanf("%f", &cie);
    if (cie < 0 || cie > 100) {
        printf("Invalid CIE input! CIE must be between 0 and 100.\n");
        i--;  // Repeats current student's entry due to invalid input
        continue;
    } else if (cie < 75) {
        printf("CIE is less than 75. You are not eligible. Better luck next time!\n");
        continue; // Skip to the next student
    }
```

Call out function to get marks:

```
// If CIE is valid, proceed to enter marks for the subjects
printf("\nEnter the Marks for Student %d: \n", i + 1);
read_marks("Mathematics", &math);
read_marks("Chemistry", &chem);
read_marks("Physics", &phy);
read_marks("English", &eng);
read_marks("Computer Science", &cs);
```

Read marks function:

```
void read_marks(const char* subject, float* mark) {
    do {
        printf("Enter the Marks of %s in Sem-I (0-100): ", subject);
        scanf("%f", mark);
        if (*mark < 0 || *mark > 100) {
            printf("Invalid input! Marks must be between 0 and 100.\n");
        }
    } while (*mark < 0 || *mark > 100);
}
```

Percent calculation:

```
average = (math + chem + phy + eng + cs) / 5;
printf("\nStudent %d's Percentage is %.2f", i + 1, average);
```

Total marks:

```
// Increment total average and valid student count
totalAverage += average;
validStudents++;

// Add to subject totals
totalMath += math;
totalChem += chem;
totalPhy += phy;
totalEng += eng;
totalCs += cs;
```

Grade calculator:

```
// Checking passing criteria
if (math < 45 && chem < 45 && phy < 45 && eng < 45 && cs < 45) {
    printf("\nSince the student does not meet the Passing Criterion, "
           "YOU ARE FAILED, better luck next time!");
    printf("\nAll the best for the next attempt!");
} else {
    if (average >= 75) {
        printf("\nThe student has bagged Distinction");
    } else if (average >= 65) {
        printf("\nThe student has bagged First Class");
    } else if (average >= 55) {
        printf("\nThe student has bagged Second Class");
    } else if (average >= 45) {
        printf("\nThe student has bagged Third Class");
    }
}
```

Batch average calculation:

```
// Calculate and print the average percentage of the batch
if (validStudents > 0) {
    float batchAverage = totalAverage / validStudents;
    printf("\n\nThe average percentage of the batch is: %.2f\n", batchAverage);

    // Calculate and print the average marks for each subject
    printf("\nAverage Marks for each subject:\n");
    printf("Mathematics: %.2f\n", totalMath / validStudents);
    printf("Chemistry: %.2f\n", totalChem / validStudents);
    printf("Physics: %.2f\n", totalPhy / validStudents);
    printf("English: %.2f\n", totalEng / validStudents);
    printf("Computer Science: %.2f\n", totalCs / validStudents);
} else {
    printf("\nNo valid students to calculate batch average.\n");
}

return 0;
```

5. Test

Test Cases:

- **Normal Scenario**: For valid marks, the program computes the average correctly, assigns grades, and prints the batch average.

- **Edge Case**: Students with marks on the boundary of grade classifications (e.g., 45, 55, 65, 75) are tested to ensure accurate grade assignment.
- **Invalid Input**: Tests confirm that the program correctly handles invalid inputs (e.g., marks out of range, low CIE) by prompting re-entry or skipping students.

Testing revealed that the program met all expected objectives, with consistent and accurate results for each scenario.

6. Implement

Where Can the Program Be Used:

This program can be implemented in:

- **Educational Institutions**: Colleges and universities for automatic grading systems.
- **Examination Boards**: Boards that manage large-scale student performance evaluations.
- **Training Institutes**: Private coaching centres that need to quickly compute results for multiple students.

Potential organizations that could implement this system include:

- **Colleges/Universities**: Particularly those dealing with large student batches, where time-efficient grading is essential.
- **Examination Councils**: Such as the CBSE, state boards, and other certification bodies.
- **EdTech Companies**: Companies like Coursera, edX, and others that need to manage results for students taking online courses.

Additional Points:

References:

Sources for the problem identification and analysis include:

- Educational feedback from teachers.
- Online articles on automated grading systems.
- Research on manual grading inefficiencies.

Future Scope:

In the future, this program can be expanded by:

- **Adding a GUI Interface**: To make the program more user-friendly for non-technical users.
- **Integration with Student Databases**: For seamless data import/export and real-time student performance tracking.
- **Advanced Analytics**: Incorporating analytics to provide deeper insights into student performance trends.

Team Cast:

B24CE1047 - Anurag Patil (Assistant Coder

B24CE1048 - Rohan Mali (Helper)

B24CE1049 - Isha Patil (Conceptualization & Publisher)

B24CE1050 - Parth Chaudhari (Coding Master)

B24CE1132 - Renuka Kulkarni (Documentation)

Presented by Everyone

THANK YOU!