📘 Transcript Compiler - Project Manual

1. How to Build & Run (Windows)

Step A: Compilation

Build the compiler by generating the C source files from your Flex/Bison code.

```
DOS
```

```
flex transcript.l
bison -d transcript.y
gcc lex.yy.c transcript.tab.c -o transcript
```

Step B: Execution

You have two ways to run the program:

Option 1: Interactive Mode (Type manually)

- Run: transcript
- Type your commands.
- When finished, press **Ctrl + Z**, then **Enter**.

Option 2: File Mode (Recommended for large data)

- Create a text file (e.g., input.txt).
- Run: transcript < input.txt

---

2. Input Syntax Reference

This compiler accepts two specific command blocks. You must follow these formats strictly.

Command 1: CALC_GPA

Calculates the GPA for a single semester and logs it to a file.

Syntax:

```
Plaintext
```

```
CALC_GPA "Semester Name" {
    COURSE_CODE : CREDIT : GRADE ;
    COURSE_CODE : CREDIT : GRADE ;
}
```

- **"Semester Name"**: Must be inside double quotes.
- **COURSE_CODE**: 3 Uppercase letters + 3 Digits (e.g., CSE110).
- **CREDIT**: Number or Decimal (e.g., 3 or 1.5).
- **GRADE**: See valid grades below.
- **Separators**: Colons : between data, Semicolon ; at end of line.

Example:

```
Plaintext
```

```
CALC_GPA "Fall 2024" {
    CSE110 : 3.0 : A+ ;
    MAT101 : 3.0 : B ;
    ENG101 : 1.5 : A- ;
}
```

Command 2: CALC_FINAL_CG

Calculates the Cumulative GPA (CGPA) based on semester summaries.

Syntax:

```
Plaintext
```

```
CALC_FINAL_CG {
   SEMESTER [Num] : GPA [Val] : CREDITS [Val] ;
}
```

Example:

```
Plaintext
```

```
CALC_FINAL_CG {
   SEMESTER 1 : GPA 3.50 : CREDITS 12.0 ;
   SEMESTER 2 : GPA 3.80 : CREDITS 15.0 ;
}
```

---

3. Valid Data Formats

Your compiler enforces strict data validation using Regular Expressions (Regex).

Data Type	Valid Examples	Invalid Examples (Will cause errors)
Course Code	CSE110, EEE201, ARC505	cse110 (lowercase), CS101 (too short), CSE-101 (dashes)
Grades	A+, A, A-, B+, B, B-, C+, C, D, F	E, A++, Fail
Credits	3, 4.0, 1.5	Three, -1

---

4. Operational Applications & Features

This section explains **what the project does** and **how it fulfills your teacher's**

requirements. Use these points during your viva/presentation.

Feature A: Academic Status Analysis (Complex Function)

- **Operation:** The compiler doesn't just calculate numbers; it evaluates the student's standing.
- **Logic:** It uses the analyze_performance() function to check if the GPA qualifies for "Summa Cum Laude," "Dean's List," or "Probation."
- **Teacher Requirement:** *Complex Function*.

Feature B: Permanent Record Keeping (File Write/Update)

- **Operation:** Every time a semester is calculated, the result is permanently saved.
- **Logic:** The system opens academic_log.txt in **Append Mode**. It never overwrites old data; it keeps a history of every run.
- **Teacher Requirement:** *File writing-update*.

Feature C: Grade Distribution Statistics (Loop)

- **Operation:** The system counts specific performance metrics (e.g., "How many 'A' grades did I get?").
- **Logic:** It iterates through an internal array (grade_history) using a for loop to generate statistics at the end of the report.
- **Teacher Requirement:** *Using Loops*.

Feature D: Cumulative Degree Auditing

- **Operation:** Students can input their summary data (Semester 1, Semester 2, etc.) to see their final graduation CGPA.

---

5. Troubleshooting

If you see syntax error or Unknown Character, check these common mistakes:

1. **Missing Semicolon:** Did you forget ; at the end of a course line?
2. **Lowercase Letters:** cse110 is invalid. You must use CSE110.
3. **Typos in Keywords:** CALC_gpa is invalid. Use CALC_GPA.
4. **File Not Found:** If using < input.txt, make sure the file is in the *exact same folder* as your transcript.exe.