Compiler Construction → Project Phase 1

Objective

The goal of Phase 1 is to design and implement a **lexical analyzer** for a **Mini C++-like language** using **Flex**. This lexical analyzer must recognize tokens, handle errors, and produce meaningful tokenized output. Each student must demonstrate **individual originality** in their project.

Project Description

You are required to implement a **scanner (lexical analyzer)** using **Flex** that performs the following tasks:

1. **Tokenization**

- Recognize and classify tokens into:
- Identifiers
- Numbers (integers, floats, exponential forms)
- Keywords (defined individually by each student, see below)
- Operators
- Punctuations
- Strings and Characters

Each token must be printed in the format:
`Line <n>: <TOKEN_TYPE> → <lexeme>`

2. **Line Tracking**
   Report the line number for each token.

3. **Error Handling**
   - Invalid tokens must be reported as errors with their line numbers.
   - Example:
   - `Line 7: ERROR → @salary (invalid identifier)`

4. **Uniqueness Requirement**
   - Each student must **choose at least 15 keywords of their own choice** (e.g., `hum`, `tum`, `Wapas`, `agar`, `output<-`, `magar`, `ucp`).
   - Students must also define **at least 3 operators** and **2 punctuations** from the provided samples or their own imagination.
   - Students must write their **own Mini C++ test program** (minimum 20 lines) using their **chosen keywords, operators, and punctuations**.
   - Projects with identical test code, identical regex, or identical outputs will not be accepted.

   Sample tokens:
   - Keywords: `Adadi, Agar, Wapas, Mantiqi, output<-`

- Operators: `=, +, &&`
- Punctuations: `{, ::`

5. **Documentation**

    - Provide a **table of regex definitions** for your tokens.
    - Draw **finite automata (FA)** diagrams for Identifiers and Numbers (hand-drawn or digital).
    - Include a brief explanation of your **choice of 15 keywords** and why you designed them.

Deliverables

1. **Document (PDF)**

    1. Regex definitions (in table form).
    2. Transition diagrams for Identifiers and Numbers.
    3. Explanation of chosen 15 keywords + operators + punctuations.

2. **Source Code**

    i. Flex file (`scanner.l`).
    ii. Unique sample Mini C++ test program (minimum 20 lines).

3. **Outputs**

    i. Token output file (`tokens.txt`).
    ii. Error log (if applicable).

4. **Demo Video (5 minutes max)**

    i. Demonstrate running your scanner on your unique program.
    ii. Explain how your regex works for at least 2 tokens.
    iii. Highlight one error-handling example.

Rubrics (10 Marks)

Criteria	Marks	Description
Regex Definitions & Correctness	2	Regex table is complete, accurate, and handles edge cases.
Flex Implementation (Tokenization)	2	Scanner correctly recognizes tokens and prints them with line numbers.
Error Handling	1	Invalid tokens are reported with line numbers.
Uniqueness & Creativity	3	Student-designed 15 keywords + operators + punctuations, plus unique sample program.
Diagrams (FA for Identifier & Number)	1	Clear, correct diagrams provided.
Demo & Explanation (Video& Viva)	1	Student demonstrates execution and explains regex.

Important Note on Individual Contribution

- Each student must **design their own set of 15 keywords**, **3-5 operators**, and **2-4 punctuations**, **Identifiers**, **Numbers.**
- Each student must create their **own sample test program** (at least 20 lines).
- Identical submissions will not be accepted.
- A short viva/demo will verify authorship.