CS4031 – Compiler Construction
Fall 2025

Total Marks: 100 (Weightage 10)
You need to work as a Group of 3

Semester Project: Mini Language Compiler – Open-Ended Design

Objective:

To design and implement a small compiler for a *domain-specific mini language* of your own choice. The project should demonstrate understanding of the six major phases of compiler construction, from lexical analysis to code generation, supported by handwritten specifications and a working implementation.

Project Requirements

1. Define Your Own Mini Language

Each group will propose and document a small custom scripting language designed for a focused purpose, such as:

- Numerical pattern generation (e.g., Fibonacci-like, factorial-based, custom arithmetic logic).
- Simple game scripting (e.g., movement or scoring rules).
- Basic data manipulation (e.g., string or array processing).
- Text-based story scripting or dialogue flow (mini DSL).
- Mathematical or logical rule engine (e.g., truth table evaluator, matrix operations).

Each team must **write their own language specification**, including:

- Syntax (BNF/EBNF grammar)
- Lexical rules (tokens, identifiers, keywords)
- Semantic rules and type system
- Example input and expected output

2. Demonstrate All Six Phases of Compilation

Students must demonstrate **complete workflow** with proper artifacts:

1. **Lexical Analysis** – token definitions and DFA construction (hand-drawn).
2. **Syntax Analysis** – grammar rules, parse tree or derivation steps.

3. **Semantic Analysis** – symbol table construction and type checking rules.
4. **Intermediate Code Generation** – three-address code or intermediate representation.
5. **Optimization (Basic)** – constant folding, dead code elimination, etc.
6. **Code Generation** – executable output or pseudo-code interpreter.

3. Implementation

- Implement the compiler in **any language of choice** (Python, C++, or Java preferred).
- Include a **simple UI or command-line interface** that lets the instructor test code snippets interactively.
- The compiler should **accept an input file in the designed mini language** and produce output accordingly.

4. Deliverables (final submission)

Each group must submit:

- **Handwritten design documents** for lexical, syntax, and semantic phases (charts, tables, parse trees, etc.) For **lexical**, **syntax**, and **semantic** phases each team must submit one **handwritten** artifact (scan/photo acceptable) DFA/transition table or regex grouping, at least two parse-tree derivations, and a sample symbol-table fill-in with scope example.
- **Printed code** with annotations for each compiler phase. Git repository (or zip) with source code and commit history.
- **Demonstration and viva voice** demonstration of their compiler executing at least 3 unique test cases.
- **Short reflection (1 page)**: what you learned, what you would improve.