Project Proposal

Course Code: CSE-314
Course Title: Compiler Design Lab
Topic Name: Mini Programming Language
Section: N2
Batch: 64
Dept. of **CSE**
Daffodil International University

Submitted by:

Name	ID
A.S.M Monjurul Islam Sajjad	0242310005101201
Md. Meherab Hossain	0242310005101200
Aovinondon Dey	0242310005101231
Tasnima Zorin Mayesha	0242310005101704
Md. Shihab Ar Rafi	0242310005101721

Submitted to:
Name: Tamanna Sultana
Lecturer
Dept. of **CSE,DIU.**

Date of submission: 23/07/25

Mini Programming Language

Introduction

The purpose of this project is to design and implement a basic interpreter/compiler for a mini programming language using Flex (Fast Lexical Analyzer Generator) and Bison (GNU Parser Generator). This language will support essential programming constructs, including variable declarations, arithmetic expressions, control structures (such as if-else and loops), and basic input/output.

This project will provide insights into the fundamental concepts of compiler design, including lexical analysis, syntax analysis, parsing, and symbol tables.

Objectives

- To design a small, yet functional, programming language syntax and semantics.
- To use **Flex** to perform **lexical analysis** (tokenization of source code).
- To use **Bison** to perform **syntax analysis** (parsing the token stream).
- To construct a symbol table for variable management.
- To generate output or interpret commands based on the parsed tree.
- To explore basic error handling in the compiler frontend.

Scope

The proposed mini language will include the following features:

- **Data Types:** Integer and float variables
- **Operators:** +, −, *, /, %, =, ==, !=, <, >, <=, >=
- **Control Structures:** `if`, `else`, `while`, `for`
- **Input/Output:** `read()`, `print()`
- **Comments:** Single-line comments starting with `//`
- **Variables:** Declaration, assignment, and use in expressions

Tools and Technologies

- **Flex** – For lexical analysis (token generation)
- **Bison** – For syntax parsing (grammar rules and AST generation)
- **C/C++** – As a base language for implementation
- **GNU Compiler Collection (GCC)** – For compiling the generated code

Methodology

1. **Language Design**
   Define the grammar and syntax rules of the mini language.
2. **Lexical Analysis with Flex**
   - Tokenize keywords, identifiers, constants, operators, etc.
3. **Parsing with Bison**
   - Write context-free grammar (CFG) rules.
   - Build parse trees and evaluate expressions.
4. **Symbol Table Management**
   - Store variable names and values.
   - Perform type checking.
5. **Execution/Interpretation**
   - Generate intermediate code or execute statements directly.
6. **Testing and Validation**
   - Write test programs to validate features and error handling.

Expected Output

- A working interpreter or compiler for the designed mini language.
- The ability to parse and execute a source code file with valid syntax.
- Display appropriate errors for invalid syntax or semantic issues.

Conclusion

This project will enhance our understanding of compiler design and language processing. By building a mini programming language from scratch, we will gain hands-on experience with lexer and parser generators and learn how source code is translated into machine-understandable actions.