COP7001 : Systems
Programming Labs

2025-2026

Contents

1 Lab 2B: Parser with Flex and Bison

Week 3 Marks: 10

1.1 Problem Statement

1.2 Functional Requirements

1. Tokenization with Flex
2. Grammar with Bison
3. AST construction
4. Support for variables, `if/else`, `while`
5. Support for loops
6. Expression evaluation

1.3 Non-Functional Requirements

Clear grammar, error messages, no crashes.

1.4 Optional Extensions

Script files, error recovery, functions.

1.5 Deliverables

Source code, grammar specification, tests, 3–4 page report.

1.6 Sample Grammar

I have listed below a sample grammar whcih you are free to enhance/modify to suit the language that your are designing. Couple of things that you should take care of:

- Spaces, tabs, and newlines must be ignored except as token separators.
- A variable must be declared using var before use.
- Declarations may optionally include an initializer
- Define and implement lexical rules precedence
- Comments can be present and must be handled

```
program         ::= statement_list ;

statement_list  ::= statement* ;

statement       ::= variable_decl
| assignment
| if_statement
| while_statement
| block ;

block           ::= '{' statement_list '}' ;

variable_decl   ::= "var" IDENTIFIER ('=' expression)? ';' ;

assignment      ::= IDENTIFIER '=' expression ';' ;

if_statement    ::= "if" '(' expression ')' statement ("else" statement)? ;

while_statement ::= "while" '(' expression ')' statement ;

expression      ::= equality ;

equality        ::= comparison (("==" | "!=") comparison)* ;

comparison      ::= term (("<" | ">" | "<=" | ">=") term)* ;

term            ::= factor (("+" | "-") factor)* ;

factor          ::= unary (("*" | "/") unary)* ;

unary           ::= ("+" | "-") unary
| primary ;

primary         ::= INTEGER
| IDENTIFIER
| '(' expression ')' ;
```