جامعة
الملك سعود
King Saud University

CSC 340 |Programming Language and Compilation

Programming Project Phase#1 : Using Lex

Students	
Name	***ID***
Aliyah Aljarallah	443201214
Shoug alsaleem	443200641
Shahad aloushan	443201080
Arwa Alfarhood	441200604

Assistance and Collaboration:
I received technical guidance from ChatGPT (OpenAI) during the debugging and refinement of my Lex file. The support primarily involved clarifying rule precedence in Lex, handling invalid identifiers (e.g., those starting with digits or ending with underscores), and enhancing the accuracy of column-based error reporting. All implementation and testing were carried out independently.

Design Choices:

1. Error Handling Priority:
   - Invalid identifiers (e.g., '2n', '_x', 'name_') are captured using custom patterns placed above the valid identifier and number rules to ensure Lex matches them first.
   - Unrecognized symbols (e.g., '?', '@', etc.) are handled using a fallback '.' rule, which reports the symbol, line number, and column.
2. Line and Column Tracking:
   - I added tracking for both 'line' and 'column' positions to provide precise error reporting.
   - After each token match, the column counter is updated by 'yyleng' to reflect the token's length.

3. Whitespace and Comments:
   - Tabs, spaces, and newlines are handled explicitly, and comments starting with '--' are skipped without generating tokens.
4. Token Output Format:
   - Each token is printed in the required format, e.g., 'IDENT var', 'NUMBER 100', 'ADD', etc., and matches the token list provided in the assignment.
5. Token Limitations:
   - Identifiers are limited to a maximum of 8 characters, as per the TINY language specification.

Feedback and Comments:
This assignment offered a practical experience with Lex and demonstrated the application of regular expressions in the construction of a custom lexer. Addressing error cases and debugging Lex patterns presented significant challenges; however, the experience proved to be rewarding. The project enhanced my understanding of the critical nature of rule ordering and pattern specificity in the realm of lexical analysis. I recommend the expansion of the official test program in future assignments to encompass a broader range of edge cases for comprehensive validation, including but not limited to consecutive invalid tokens, embedded comments, and complex nesting of expressions.

Final Note:
The test cases were carefully selected and reviewed. Based on testing so far, the analyzer appears to behave as expected, including terminating upon detecting lexical errors. However, additional edge cases may still exist, and further testing is encouraged to ensure complete robustness.