CSL302: Compiler Design

Assignment-1

Due Date: 3-September-2025

Overview of CSL302 Assignments:

In this course, you will write a compiler for a variant of object-oriented programming language, which is similar to C++/Java. Note that It is not an exact match to any of those languages. The feature set has been trimmed down and simplified to keep the assignments manageable.

Logistics on Submissions:

- Implement your solution using the lexical analyzer tool, for example flex.
- The assignments have to be solved in a team. The team size should be maximum 2. You can enter your team details here **by 27-Aug-2025.**
- Prepare your submission in a zip file and name it as <ROLL NO1_ROLL NO2>.zip. If the team size is other than two, name your submission accordingly.
- Your submission should include a README file containing the instructions to execute your program(s).
- Upload your assignment in the canvas portal.
- Note that the weightage of each assignment will be different.
- It is sufficient if one member of the team submits the file.
- Any form of plagiarism is strictly prohibited; all the text must be your own, and all the references must be cited.

Assignment Problem Statement:

In this assignment, you are required to implement the lexical analyzer for a variant of object-oriented programming, which has the following features.

Keywords:

void int double bool string extends implements for while ReadInteger ReadLine class interface if else return null this break New NewArray Print

Identifiers:

An identifier is a sequence of letters, digits, and underscores, starting with a letter. Note that our programming language is case-sensitive, e.g., if is a keyword, but IF is an identifier; binky and Binky are two distinct identifiers. Identifiers can be at most 31 characters long.

Whitespaces:

Whitespace (i.e. spaces, tabs, and newlines) serves to separate tokens, but is otherwise ignored. Keywords and identifiers must be separated by whitespace or a token that is neither a keyword nor an identifier.

Constants:

Booleans: A boolean constant is either true or false.

Integers: An integer constant can either be specified in decimal (base 10) or hexadecimal (base 16). A decimal integer is a sequence of decimal digits (0-9) . A hexadecimal integer must begin with 0X or 0x and is followed by a sequence of hexadecimal digits. Hexadecimal digits include the decimal digits and the letters a through f (either upper or lowercase). Examples of valid integers: 8, 012, 0x0, 0X12aE.

Doubles: A double constant is a sequence of digits, a period, followed by any sequence of digits, maybe none. Thus, .12 is not a valid double but both 0.12 and 12 . are valid. A double can also have an optional exponent, e.g., 12.2E+2 For a double in this sort of scientific notation, the decimal point is required, the sign of the exponent is optional (if not specified, + is assumed), and the E can be lower or upper case. As above, .12E+2 is invalid, but 12.E+2 is valid. Leading zeroes on the mantissa and exponent are allowed.

Strings: A string constant is a sequence of characters enclosed in double quotes. Strings can contain any character except a newline or double quote. A string must start and end on a single line, it cannot be split over multiple lines:

Operators:

+ - * / % \ < <= > >= = == != && || ! ; , . [] [ ] ( )

Comments:

A single-line comment is started by // and extends to the end of the line. Multi-line comments start with /* and end with the first subsequent */ . Any symbol is allowed in a comment except the sequence */ which ends the current comment. Multi-line comments do not nest.