Assignment on Compiler Design (CSE_3151)

Total Point : 10 Marks
Due Date: Assignment Submission week (Demonstration + Project Report Submission) [10th-17th April, 2025]

Objective:

- Design, implement and document an entire compiler chain [Lexical Analyzer, Parser, Intermediate Code Generator and Code Generator] for any one option listed below.
    - **Option 1**: programming language described on the following pages.
    - **Option 2**: programming language as per your choice.

- The compiler will produce a listing file which lists and numbers each line, interleaved with any error messages following the source file. Each error message should refer to a line number. Even if errors are present in the source code, your compiler should continue to analyze the input source code. Target code will be generated only if there are no errors.

- The entire project is due to be in the assignment week (10-17 April, 2025) of the semester. Students are required to demonstrate the compiler. Students must submit
    - Synopsis as a .pdf file: Due Date – 22nd February, 2025
    - Source code of the compiler and Presentation PPT (.pdf form) as a zip file . Due Date – 10-17TH April, 2025

DESCRIPTION OF THE PROGRAMMING LANGUAGE

[If you choose option 1 only]

Lexical Units and Spacing

The lexical units of a program are identifiers, reserved words, numbers, strings, and delimiters.

A delimiter is any one of the following special characters:
( ) [ ] ; : . , * - + / < = >
or one of the following compound symbols:
<> := <= >=
Spaces may be inserted freely with no effect on meaning between lexical units. At least one space must separate adjacent identifiers or numbers. The end of a line is equivalent to a space except (1) it terminates comments and (2) is illegal in quoted character strings. Thus, each lexical unit must fit on one line.

IDENTIFIERS

Identifiers (names) consist of a letter followed by a sequence of letters or digits. Upper and lower case are considered equivalent. An identifier may be any length but must be distinguishable within the first 32 characters. Thus, the two names
a11111111111111111111111111111112222 and
a11111111111111111111111111111119999 are not distinguishable.

NUMBERS

The only kind of number is the integer. An integer is a sequence of digits. Maximum and minimum values of integers are determined by our implementation.

STRINGS

A character string is a sequence of characters prefixed and terminated by the apostrophe character. The backslash character (\) acts as an escape. When followed by an **n** or a **t**, the backslash denotes a new line (CR) or tab. When followed by another character it denotes that character. Thus \' denotes the single quote. For example, the string DON'T would be represented as 'DON\'T'.

COMMENTS

A comment starts with an exclamation point (!) and is terminated by the end of a line. It may appear following a lexical unit or at the beginning of a program.

!This is a comment

RESERVED WORDS

The identifiers listed below are reserved words and may not be declared by the programmer:

and, array, begin, integer, do, else, end, function, if, of, or, not, procedure, program, read, then, var while, write

Syntax :

Production rules are as follows:

program -> program id;
type -> standartype | array[num]
standard~type -> integer
statement -> variable assignop expression
| procedure_statement
| compound_statement
| if expression then statement else statement
| if expression then statement
| while expression do statement
| read_statement
| write_statement
arguments -> (parameter_list) | ( )
read_statement -> read (identifier_list)
identifier__list -> variable,identifier_list | variable
write_statement -> write(outputlist)
outputlist -> outputitem |outputitem,outputlist
outputitem -> string | expression_list
factor -> id[expression]

In addition: Students can think about different types of operators, Arrays.