Assignment 02 – Recursive Descent Parser Revision **(DUE: Fri March 7, 2025)**

The initial description of a simplified "LISP-like" language (syntactically, at least) provided earlier in the "Simple RDP Example" module on Brightspace is repeated below.

Consider the following tabular description of a "Finite State Machine Lexical Analyzer" that recognizes each of the following lexical elements and generates their corresponding tokens.

LP - Left Parentheses: 1, RP - Right Parentheses: 2, ID - Identifier: 3, IL - Integer Literal: 4, SL - String Literal: 5

space new line

	(	)	"	+-	1234567890	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_		
Start	Start{1}	Start{2}	SL	IL	IL	ID	Start	Start
SL	SL	SL	Start{5}	SL	SL	SL	SL	SL
IL	Start{4&1}	Start{4&2}	SL{4}	IL{-1}	IL	ID{4}	Start{4}	Start{4}
ID	Start{3&1}	Start{3&2}	SL{3}	IL{3}	ID	ID	Start{3}	Start{3}

Consider this accompanying grammar for a language with a simple syntax that utilizes the same lexemes. Note that, for the sake of clarity and consistency, the non-terminals have been named to correspond to the lexical elements of the preceding question.

<Unit>	Compilation Unit
<SEs>	Symbolic Expressions
<SE>	Symbolic Expression
<SA>	Symbolic Atom
<LA>	Literal Atom
<L>	List

<Unit> ➔ <SE> <SEs>
<SEs> ➔ <SE> <SEs> |

<SE> ➔ <SA> | <LA> | <L>

<SA> ➔ <ID>

<LA> ➔ <IL> | <SL>

<L> ➔ (<SEs>)

Example Source: `FE_01.txt`	Corresponding Lexical Tokens: `FE_01.tkn`

`( Item 17 "Hello" )`
`( ) I -23 ""`
`( -45 )`
`"unfinished`
`now finished"`
`And1and2and3`
`+3`
	`1 3 4 5 2 0`
	`1 2 3 4 5 0`
	`1 4 2 0`
	`0`
	`5 0`
	`3 0`
	`4 0`
	`0`

The original Java program, **`FERDP.java`**, also provided earlier in the "Simple RDP Example" module on Brightspace, serves as a "Recursive Descent Parser" for this language, and contains a modification (on lines 46 and 48) to have it completely ignore all of the zero tokens. Although this modification allows the program to produce correct results, it defeats the purpose of having the zero tokens in the input; which is to serve as markers for line breaks thus allowing for a precise correlation between the token stream and the original character stream.

In class, I explained all of this and then proceeded to modify the program by "commenting out" lines 46 and 48 and incorporating the use of an additional method, **`next`**, designed to return the index of the next non-zero token given the index of a token in the stream. This attempt is not completely thought out and thus is incomplete and ineffective. However, I do think it is an incremental step toward a full and workable solution. This version of the program is provided in the "AS02 – RDP Revision" module on Brightsapce.

For this assignment you are to think the approach out further and continue the modifications of the program to arrive at a full and workable version of the program. As decided in class, I will accept submissions in the names of individuals, or self-selected teams of students who have collaborated together on this assignment. Let's remember that the purpose of assignments is "to learn" and not merely to check off some requirement, so as to earn a grade. Thus, not only do I expect the team formation to be reasonable, but I also place the responsibly for illustrating the results and for specifying the contributions of all on the submitters.

Each submission for this assignment is to consist of:

- The revised source program, still named **`FERDP.java`**.

- A MS Word readable document file, to be named **`AS02.docx`**, that serves as a concise report on the work being submitted. In addition to the standard "identifying information" for the report, this document must provides the submitter(s) the opportunity to describe, explain and illustrate what has been done and what has been accomplished. You may find it useful to imagine that I will not be compiling and executing your program on test cases, and so it is left up to you to provide such evidence.

  I think a particularly useful way to present what you have done would be to prepare a "side-by-side" comparison of the given version of the program with the submitted version. I commonly use MS Word's two-column capability to prepare such a comparison, using color to highlight differences, with red and yellow used on the original to indicate deletions and modifications respectively, and green and blue used on the revision to indicate modifications

and additions respectively. I further think that this approach has merit given the name of this assignment.

This document must also contain a section which provides specific clarifications on the contributions of each member of the team.

Additional and Optional

Additionally and optionally, I encourage those of you who think you have successfully completed this assignment (by evolving the given program into a fully workable one), to prepare and propose a set of "test cases" to be used to test the correctness of the revised program.

For each "test case" in the set, there should be a **`*.txt`** file and a corresponding **`*.tkn`** file, and then each test case should be illustrated in a MS Word readable document file, to be named **`AS02AO.docx`**.

Note that the test set you prepare and submit should not include any of the already given examples, **`FE_01.txt`** and **`FE_02.txt`**, as including these would add nothing new. Instead I recommend that you consider a methodical approach that presents a more thorough and progressively more involved sequence of test cases. That is, going from simple to more complex up to some point.

For this A&O portion of the assignment, please prepare and submit a single "ZIP file" (named **`AS02AO.zip`**) that contains the **`AS02AO.docx`** file and the **`*.txt`** files and corresponding **`*.tkn`** files.

Good luck,
P.M.J.