German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail

Compilers, Spring Term 2023
Assignment 1

Due on May 5, by 23:59

In this assignment, you will need to do some (simple) research and read about *definite clause grammars*.[1] Using Prolog, write a parser for a definite clause grammar of *Java-Light*. Java-Light is a fragment of Java consisting of non-empty semi-colon-separated sequences of *some* grammatical Java statements. In particular, we restrict ourselves to the following Java statements.

a) Assignment statements.

- The left side of the assignment is a Java identifier and the right side is an arithmetic expression involving only the binary operators `+`, `-`, `*` , `/`, and `%`. The expression can have parenthesized sub-expressions and its atomic sub-expressions are identifiers or unsigned `int` literals.
- For example, the following is acceptable

```
counter1 = counter1 + (x / y - 21) % _w2;
```

  whereas the following is not

```
counter1 = 1counter ++x / y - 21) % _w2
```

b) Conditional Statements.

- Conditional statements are restricted to valid Java `if` and `if-else` statements, nested to any depth, and with bodies which are single statements.
- The conditions in these statements are simple relational expressions involving one of the operators `==`, `!=`, `<=`, `<`, `>=`, and `>` flanked by arithmetic expressions (as restricted above).

c) Loops.

- Loops are restricted to valid Java `while` loops, nested to any depth, with conditions as restricted above, and with bodies which are single statements.

Thus, the following is valid in Java-Light. (Line breaks and tabs added for readability.)

```
counter = x + y;
while (counter <= w - 1)
  while (counter != y)
    counter = counter + x + 5;
```

[1] Check out `https://en.wikipedia.org/wiki/Definite_clause_grammar` as a starting point.

```
if (counter > w + 2)
  if (counter > x)
    counter = y;
  else
    if (counter > y)
      counter = x;
w = y / x;
```

Your grammar should contain *no* ε-rules and should have the symbol s as its start variable.

A successful parse should result in building a parse tree for the input; make sure your parser generates such a tree. We linearly represent a parse tree as follows:

a) l, where l is the label of a leaf.

b) $p(l1, l2, \ldots, ln)$, where p is a label of a parent node and li is the i^{th} sub-tree thereof, where left-to-right order is assumed.

Submission Guidelines

- This is an *individual* assignment.
- You are required to submit a single .pl. The name of the file should be *only* your ID (Example: 46-1234.pl). We shall soon announce how to submit.
- The soundness and completeness of your grammar will be auto-tested. You need to make sure that queries are of the following format `s(T,[c,o,u,n,t,e,r,=,0,;],[]).` run correctly.