Mini Assignment 3 – Semantic Evaluation

Total Points = 10

Instructions:

1. You must use SWI Prolog to implement this problem.
2. You must submit this assignment on Canvas by clicking the title link for the assignment. Make sure your programs compile (without any compiler errors). You will not receive credit if your program for a problem does not compile. If you are unable to complete any of the programs, submit the parts that work (with no compiler errors) for partial credit.
3. Your grade will be based on meeting the requirements, functionality (does the program do what it is supposed to do), readability (is the code nicely indented and formatted), and understandability (are the literals meaningful and is the code well documented with appropriate comments). You must incorporate all the good programming practices and styles.

Deliverables:

1. You should **submit one .pl file** named **miniAssignEval.pl.** This file should contain your code for the problem in the assignment for the grader to run. This file will only contain the code and not the sample runs.
2. You should also submit **one PDF file** named **miniAssignEval.pdf**. This PDF file should contain code to the solution of the problems (in **text format only, no snapshot** will be accepted) and your program's sample runs with output (in **text format only, no snapshot** will be accepted). Make sure to paste your code correctly (with **correct indentation**). If you have any text for the grader to read before grading, you can include it at the top of this file as comments or in canvas notes.

Q1. Program an evaluator for the expression rules whose definition is given below (it is part of the language given assignment 3).

E in Arithmetic Expression
I in Identifier
N in Number

E ::= E + E | E - E | E * E | E / E | (E) | I | N
I ::= x | y | z | u | v
N ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

As to create an evaluation predicate you need an environment, consider it to be of the form [(<var1>, <value>), (<var2>, <value>), (<var3>, <value>)], for example for an environment where x = 2 and z = 5, it is [(x,2), (z,5)].

To evaluate an expression, write two predicates:

1. **expr**(Tree, Tokens, []) where **Tokens** of expression to be evaluated as inputs and generates a parse tree as **Tree**.
2. **eval_expr**(Tree, Env, Result), where **Tree** is the parse tree generated using **parse**, **Env** is the environment in the format described above. **Result** holds the result after evaluation.

For example, your program should accept the following query:

?- **expr**(T, [1, +, x, +, 2], []), **eval_expr**(T, [(x,4)], R), and return R = 7.

?- **expr**(T, [1, +, 6, +, 2, *, 0], []), **eval_expr**(T, [], R), and return R = 7.

Remember to keep the correct operator precedence (/, *) before (+, -) and associativity.

You can use the tabling feature in SWI prolog to eliminate left recursion.