CS3513
Programming Languages
HomeWork 1 (Individual Assignment)

The following Context-free grammar generates Boolean expressions, with atomic elements, operators, precedence and associativities that you should recognize from the lectures.

```
E -> E or T
  -> E nor T
  -> E xor T
  -> T
T -> F and T
  -> F nand T
  -> F
F -> not F
  -> P
P -> ( E )
  -> i
  -> true
  -> false
```

1. Determine whether this grammar is LL(1) or not. Justify your answer.
2. If the grammar is not LL(1), transform it into an equivalent grammar that is LL(1).
3. Build a parse table from your resulting grammar in Part 2.
4. Using your parse table from Part 3, show how the following input string is parsed:

   `true nand (false xor i) or not i and not false nor i`

5. Using the table lookups from Part 4, build the derivation tree (top-down, left-most derivation) for the string parsed in Part 4.

6. Write (pseudo) code for Recursive Descent parser for the above grammar (LL(1) version). Include "Write()" statements to build the Derivation Tree in bottom-up fashion.

7. Show the sequence of productions produced by your recursive descent parser, for the following input string:

   `(i nor false) and true or i nor not i and true xor i`

8. Show the corresponding derivation tree.

9. Modify the code for the above Recursive Descent parser (From Q6), so that it generates the derivation tree for the original grammar.

10. Show the sequence of productions produced by your recursive descent parser, for the following input string:

```
(i nor false) and true or i nor not i and true xor i
```

    Show the corresponding derivation tree.

11. Modify the same code so it will generate the Abstract Syntax Tree. The String-to-Tree Transduction grammar is as follows:

E -> E or T	=> 'or'
-> E nor T	=> 'nor'
-> E xor T	=> 'xor'
-> T	
T -> F and T	=> 'and'
-> F nand T	=> 'nand'
-> F	
F -> not F	=> 'not'
-> P	
P -> (E)	
-> i	=> 'i'
-> true	=> 'true'
-> false	=> 'false'

12. Show the sequence of calls to Build_tree that are carried out by your new recursive descent parser, for the same input string as in Q10. Show the corresponding Abstract Syntax Tree.