LL(1) Grammars & LL(1) Parsing

Exercise 1

Consider the following grammar for balanced parentheses:

```
S ::= B EOF
B ::= ε | B ( B )
```

Question 1.1

Compute NULLABLE, FIRST and FOLLOW for each non-terminal.

Question 1.2

Build the LL(1) parsing table for the grammar. Is the grammar LL(1)?

Question 1.3

Build the LL(1) parsing table for the following grammar. Is the grammar LL(1)?

```
S ::= B EOF
B ::= ε | ( B ) B
```

Question 1.4

Run the LL(1) parsing algorithm on the following input strings. Show the derivation you obtain, or the derivation up until the error.

(( ) ( )) EOF

( )) (EOF

Question 1.5

Show that the second grammar describes the language of balanced parenthesis.
To do so, show that:

1) Every parse tree of the grammar yields a balanced parenthesis string.
2) For every balanced parenthesis string, there exists a parse tree.

Exercise 2

Question 2.1

Build a LL(1) grammar for expressions consisting of variables, infix binary addition and multiplication (with usual priorities), and parentheses. Note that we do not care about associativity of binary operations at the level of the parse tree.

Build the LL(1) parsing table.

Question 2.2

Parse the following input strings using your parsing table. Show the derivations and parse trees.

x + y * (z + x) EOF

x * y * z EOF

((x)) EOF

Exercise 3

Consider the following grammar for roman numerals 0 to 9:

S ::= N **EOF**
N ::= **i** A | I_3 | **v** I_3 | ε
A ::= **v** | **x**
I_3 ::= I_2 **i** | ε
I_2 ::= I_1 **i** | ε
I_1 ::= **i** | ε

Question 3.1

Is the grammar ambiguous? In other words, do there exist multiple parse trees for the same words ?

Question 3.2

Is the grammar LL(1)? To motivate your answer, build an LL(1) parsing table. If there are conflicts, discuss why each conflict appears.

Question 3.3

Assuming the above is not LL(1), build a LL(1) grammar for the same language. Show it is LL(1) by building a parsing table.