Exercise 1

Show that the following grammar is `LL(1)`. To do so, compute `Nullable`, `First` and `Follow` for `S` and `B` and build the `LL(1)` parsing table.

```
S -> B EOF
B -> ε | ( B ) B
```

Exercise 2

Show that the following grammar is ambiguous. To do so, find two different parse trees that correspond to the same input string.

```
S -> E EOF
E -> E + E | E * E | E E | ( E ) | V
V -> x | y
```

Exercise 3

Consider the following grammar:

```
S  -> E EOF
E  -> E + T | T
T  -> T O F | F
O  -> * | ε
F  -> ( E ) | V
V  -> x | y
```

Question 1

Is the language defined by this grammar the same as the one defined by the grammar from the previous exercise? If so, prove it is indeed the case by outlining a recursive procedure to convert parse trees from one grammar to and from parse trees in the other grammar that yield the same word.

Question 2

Is the grammar ambiguous? Briefly justify your answer.

Question 3

Compute `Nullable`, `First` and `Follow` for the grammar. Is the grammar `LL(1)`? Justify your answer.

Exercise 4

Question 1

Transform the grammar from the previous exercise into a `LL(1)` grammar.

Question 2

Compute `Nullable`, `First` and `Follow` for all non-terminals of your `LL(1)` grammar.

Question 3

Build the `LL(1)` parsing table for your grammar.

Question 4

Using your `LL(1)` parsing table, parse the following three input strings:

```
x*y+y

x+yy

(x+y)x
```