Solution - Exercises on BNF, EBNF, and Parse Trees

1. Consider the following grammar:

```
<exp>       ::=   <atom> | <list>
<atom>      ::=   <digit> | <string>
<list>      ::=   ( <expr-list> )
<expr-list> ::=   <exp> <expr-list> | ε
<digit>     ::=   0 | 1   | 2 | ... | 9
<string>    ::=   a | b | c
```

What change would you make to transform it into EBNF?

```
<expr-list> ::=   { <exp> }
```

2. Given the following grammar with tokens **n**, **l**, **o**, **c**, and <G> as the start symbol:

```
<G>   ::=   <A> | <L>
<A>   ::=   n | l
<L>   ::=   o <S> c
<S>   ::=   <S> <G> | <G>
```

Convert it to EBNF.

```
<G>   ::=   <A> | <L>
<A>   ::=   n | l
<L>   ::=   o <S> c
<S>   ::=   <G> { <G> }
```

3. Write a BNF description for an identifier sequence which is a sequence of identifiers separated by commas. Assume that identifiers are denoted by the token id. In addition, translate your BNF into EBNF.

```
BNF     <id-seq>   ::=   id , <id-seq> | id
EBNF    <id-seq>   ::=   id { , id }
```

4. You are given the following grammar, with tokens **a**, **b**, **c**, **d**, and <S> as the start symbol:

```
<S>   ::=   a <L> b     | c
<L>   ::=   <L> d <S>   | <S>
```

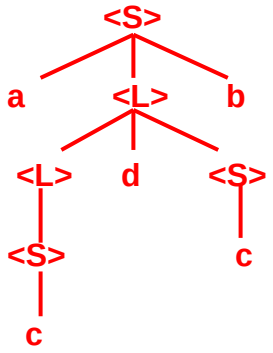
a. Give a leftmost derivation for the string **acdcb.** **Note the change**

<S> ⇒ a <L> b ⇒ a <L> d <S> b ⇒ a <S> d <S> b ⇒ a c d <S> b
⇒ a c d c b

b. Give a rightmost derivation for the same string.

<S> ⇒ a <L> b ⇒ a <L> d <S> b ⇒ a <L> d c b ⇒ a <S> d c b
⇒ a c d c b

c. Draw a parse tree for that string.



5. You are given the following grammar, with tokens **a**, **b**, and <S>, <A>, and <B> as the non-terminals <S> as the start symbol:

<S> ::= <A> <B> | ε
<A> ::= **a** <B>
<B> ::= <S> **b**

a. Give a leftmost derivation for the string **abb**. You may use S, A, and B instead of <S>, <A>, and <B> respectively.

S ⇒ A B ⇒ a B B ⇒ a S b B ⇒ a b B ⇒ a b S b ⇒ a b b

b. Give a rightmost derivation for the same string.

S ⇒ A B ⇒ A S b ⇒ A b ⇒ a B b ⇒ a S b b ⇒ a b b

6. You are given the following grammar, with tokens **a**, **b**, and <S> as the start symbol:

<S> ::= **a** <A> <B>
<A> ::= **b** <B> **b**
<B> ::= <A> | ε

a. Give a leftmost derivation for the string **abbbb** **Note the change**

S ⇒ a A B ⇒ a b B b B ⇒ a b A b B ⇒ a b b B b b B ⇒ a b b b b B ⇒ a b b b b

b. Give a rightmost derivation for the same string

S ⇒ a A B ⇒ a A ⇒ a b B b ⇒ a b A b ⇒ a b b B b b ⇒ a b b b b

c. Draw a parse tree for that string.