Example: A simple programming language grammar.

\[
G: \text{<Exp>}: = \text{<Exp>} + \text{<Exp>}
| \quad \text{<Exp>} * \text{<Exp>}
| \quad (\text{<Exp>})
| \quad \text{a}
| \quad \text{b}
| \quad \text{c}
\]

Terminal symbols!!!

\[
S = \text{a}
S = \text{a + b}
S = \text{a + b * c}
S = (\text{a + b}) * \text{c}
S = ( (\text{a + b}) )
S = \text{c(a + b)}
S = (\text{c}) + (\text{b})
S = \text{b++}
\]

\[
S \in L(G)\
\]
The empty symbol: `<empty>`
You can think of `<empty>` being defined by the implicit rule:

```
<empty> ::= ""
```
That is the `<empty>` symbol derives nothing.
Consider the grammar:

\[
G: \quad \langle A \rangle^* ::= a \langle B \rangle \mid a \\
\langle B \rangle ::= b \langle B \rangle \mid b
\]

\[
G' : \quad \langle A \rangle^* ::= a \langle B \rangle \\
\langle B \rangle ::= b \langle B \rangle \mid \langle \text{empty} \rangle
\]
Consider the following grammar fragment:

<if-stmt> ::= if <exp> then <stmt> <else-part>
<else-part> ::= else <stmt> | <empty>
<exp> ::= ...
<stmt> ::= ...

Consider the diagram below:

```
<if-stmt>
  /
 if <exp> then <stmt> <else-part>
    /
 else <stmt>
```

```
<if-stmt>
  /
 if <exp> then <stmt> <else-part>
    /
         <empty>
```
2.1 a) Let $L(G)$ be the language of all strings consisting of zero or more a’s.

2.1 i) Let $L(G)$ be the set of strings consisting of one or more a’s with a comma between each a and the next.

2.1 d) Let $L(G)$ be the set of all strings consisting of one or more digits 0 – 9.