1. For the following sub-problems, consider the following context-free grammar:

\[ S \rightarrow A\$ \]  \hspace{1cm} (1)
\[ A \rightarrow xAx \]  \hspace{1cm} (2)
\[ A \rightarrow C \]  \hspace{1cm} (3)
\[ B \rightarrow yBy \]  \hspace{1cm} (4)
\[ B \rightarrow C \]  \hspace{1cm} (5)
\[ C \rightarrow zBz \]  \hspace{1cm} (6)
\[ C \rightarrow wAw \]  \hspace{1cm} (7)
\[ C \rightarrow \lambda \]  \hspace{1cm} (8)

(a) What are the terminals and non-terminals of this grammar?

**Answer:** Terminals: \{x, y, z, w, \$\}, non-terminals: \{S, A, B, C\}

(b) Show the derivation of the string \textit{xzzx\$} starting from \textit{S} (specify which production you used at each step), and give the parse tree according to that derivation.

**Answer:**

\[
\begin{align*}
S & \rightarrow A\$ \quad \text{Rule 1} \\
 & \rightarrow xAx\$ \quad \text{Rule 2} \\
 & \rightarrow xCx\$ \quad \text{Rule 3} \\
 & \rightarrow xCzBz\$ \quad \text{Rule 6} \\
 & \rightarrow xCzx\$ \quad \text{Rule 5} \\
 & \rightarrow xzzx\$ \quad \text{Rule 8}
\end{align*}
\]

The parse tree follows directly from this derivation.

(c) Give the first and follow sets for each of the non-terminals of the grammar.

**Answer:**

\[
\begin{align*}
First(S) & = \{x, z, w, \$\} \\
First(A) & = \{x, z, w, \lambda\} \\
First(B) & = \{y, z, w, \lambda\} \\
First(C) & = \{z, w, \lambda\}
\end{align*}
\]
Follow(S) = {}
Follow(A) = \{\$, x, w\}
Follow(B) = \{y, z\}
Follow(C) = \{\$, x, y, z, w\}

(d) What are the predict sets for each production?

Answer:

\begin{align*}
\text{Predict}(1) & \quad \{x, z, w, \$\} \\
\text{Predict}(2) & \quad \{x\} \\
\text{Predict}(3) & \quad \{z, w, x, \$\} \\
\text{Predict}(4) & \quad \{y\} \\
\text{Predict}(5) & \quad \{z, w, y\} \\
\text{Predict}(6) & \quad \{z\} \\
\text{Predict}(7) & \quad \{w\} \\
\text{Predict}(8) & \quad \{\$, x, y, z, w\}
\end{align*}

(e) Give the parse table for the grammar. Is this an LL(1) grammar? Why or why not?

Answer:

\begin{center}
\begin{tabular}{c|ccccc}
    & x & y & z & w & \$ \\
\hline
S  & 1 & 1 & 1 & 1 & \\
A  & 2, 3 & 3 & 3 & 3 & \\
B  & 4, 5 & 5 & 5 & \\
C  & 8 & 6, 8 & 7, 8 & 8 & \\
\end{tabular}
\end{center}

This is not an LL(1) grammar, because there are conflicts in the parse table.

(f) Show the steps your parser would take to parse “xzxyzx$”.

Answer: The parser would not be able to parse this string, because there are conflicts in the parse table.
2. for the following sub-problems, consider the following grammar:

\[
\begin{align*}
&S \to AB\$ & (1) \\
&A \to xA & (2) \\
&A \to B & (3) \\
&B \to yzB & (4) \\
&B \to z & (5)
\end{align*}
\]

(a) What are the terminals and non-terminals of this grammar?

**Answer:** Terminals: \{x, y, z, \$\}; non-terminals: \{S, A, B\}

(b) Show the parse tree for \textit{xyzzz\$}.

**Answer:**

(c) What are the first and follow sets for each of the non-terminals of the grammar?

**Answer:**

\[
\begin{align*}
First(S) &= \{x, y, z\} \\
First(A) &= \{x, y, z\} \\
First(B) &= \{y, z\}
\end{align*}
\]

\[
\begin{align*}
Follow(S) &= \{} \\
Follow(A) &= \{y, z\} \\
Follow(B) &= \{y, z, \$\}
\end{align*}
\]
(d) What are the predict sets for each production?

**Answer:**

\[
\begin{align*}
\text{Predict}(1) & = \{x, y, z\} \\
\text{Predict}(2) & = \{x\} \\
\text{Predict}(3) & = \{y, z\} \\
\text{Predict}(4) & = \{y\} \\
\text{Predict}(5) & = \{z\}
\end{align*}
\]

(e) Give the parse table for this grammar. Is this an LL(1) grammar?

**Answer:**

\[
\begin{array}{|c|c|c|c|c|}
\hline
 & x & y & z & $ \\
\hline
S & 1 & 1 & 1 & \$ \\
A & 2 & 3 & 3 & \\
B & 4 & 5 & \\
\hline
\end{array}
\]

This is an LL(1) grammar, as there are no conflicts in the parse table.

(f) If we add the rule \( A \to \lambda \), is the grammar still LL(1)? Why or why not?

**Answer:** Let us call the new rule rule 6. We can rebuild the first, follow, and predict sets:

\[
\begin{align*}
\text{First}(S) & = \{x, y, z\} \\
\text{First}(A) & = \{x, y, z, \lambda\} \\
\text{First}(B) & = \{y, z\}
\end{align*}
\]

Note that the First set of \( A \) changed.

\[
\begin{align*}
\text{Follow}(S) & = \{} \\
\text{Follow}(A) & = \{y, z\} \\
\text{Follow}(B) & = \{y, z, \$\}
\end{align*}
\]

Note that none of the follow sets changed!
\[ \text{Predict}(1) = \{x, y, z\} \]
\[ \text{Predict}(2) = \{x\} \]
\[ \text{Predict}(3) = \{y, z\} \]
\[ \text{Predict}(4) = \{y\} \]
\[ \text{Predict}(5) = \{z\} \]
\[ \text{Predict}(6) = \{y, z\} \]

But the predict set for rule 6 is \text{Follow}(A). If we build the parse table for this new grammar, we get:

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>3, 6</td>
<td>3, 6</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which means we have a conflict: if we're expanding an $A$, and we see a $y$ or $z$, don't know whether to turn it into a $B$ using rule 3 or to remove it (turn it into $\lambda$) using rule 6. Thus, the grammar is not LL(1).