1. Give the reduced DFA for the following regular expression:

\(((a^*bcd)|(bc^*d))\)

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

\[
S \rightarrow AB \\
A \rightarrow xAC \\
A \rightarrow \lambda \\
B \rightarrow CBy \\
B \rightarrow \lambda \\
C \rightarrow z
\]

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

**Answer:** \(V_t = \{x, y, z\}\) and \(V_n = \{S, A, B, C\}\)

(b) Describe the strings are generated by this language. Is this a regular language (i.e., could you write a regular expression that generates this language)?

**Answer:** This language generates strings of the form \(x^n z^{m+n} y^m\). This is not regular because it requires a potentially unbounded amount of state to track how many \(n\)s and \(m\)s there are.

(c) Show the derivation of the string \(xxzzzy\) starting from \(S\) (specify which production you used at each step), and give the parse tree according to that derivation.

**Answer:** I will put a superscript on the production arrow \(\Rightarrow\) to denote which production was used at each step.

\[
S \Rightarrow^1 AB \Rightarrow^2 xACB \Rightarrow^2 xxACCB \Rightarrow^3 xxCCB \Rightarrow^4 \\
xxCCBy \Rightarrow^5 xxCCCy \Rightarrow^6 xxzzCCy \Rightarrow^6 xxzzCy \Rightarrow^6 xxzzzy
\]

The parse tree is as follows:
(d) Give the first and follow sets for each of the non-terminals of the grammar.

**Answer:**

First($S$) = \{x, z, $\}$  
First($A$) = \{x, $\lambda$\}  
First($B$) = \{z, $\lambda$\}  
First($C$) = \{z\}  
Follow($S$) = \{$\$\}  
Follow($A$) = \{z, $\$\}  
Follow($B$) = \{y, $\$\}  
Follow($C$) = \{y, z, $\$\}  

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

**Answer:**

Predict(1) = \{x, z\}  
Predict(2) = \{x\}  
Predict(3) = \{z, $\$\}  
Predict(4) = \{z\}  
Predict(5) = \{y, $\$\}  
Predict(6) = \{z\}  

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

This is an LL(1) grammar because there are no predict conflicts.
3. for the following sub-problems, consider the following grammar:

\[
\begin{align*}
S & \rightarrow AB \quad (7) \\
A & \rightarrow xA \quad (8) \\
A & \rightarrow xyB \quad (9) \\
B & \rightarrow zB \quad (10) \\
B & \rightarrow w \quad (11) \\
\end{align*}
\]

(a) Describe the strings generated by this language.

**Answer:** This language generates strings of the form \(x^+yz^*w\).

(b) Is this language LL(1)? Why or why not?

**Answer:** This language is not LL(1) because both productions 8 and 9 start with an \(x\). The predict sets for both would be \(x\), and there would be a conflict in the parse table.

(c) Build the CFSM for this grammar.

(d) Build the goto and action tables for this grammar. Is it an LR(0) grammar? Why or why not?

These can be built directly from the above CFSM. It is an LR(0) grammar because there are no shift/reduce or reduce/reduce conflicts (every state is either a shift state or a reduce state).
(e) If we add the production

\[ A \rightarrow y \]

to the grammar, is it an LR(0) grammar? Why or why not?

**Answer:** Consider what happens to States 3 and 4 if the production \( A \rightarrow y \) is added:

If we are in State 3 and we see a \( y \), we move to State 4. Unfortunately, State 4 has a shift/reduce conflict: we don’t know if we should reduce \( A \) or shift to see the \( z \) or \( w \).