Problem 1:
Merriam-Webster defines palindrome as:

>a word, verse, or sentence (as "Able was I ere I saw Elba") or a number (as 1881) that reads the same backward or forward

Perform the following:

1. Briefly discuss why regular expressions cannot recognize arbitrary length palindromes.

2. Write a grammar which accepts arbitrary length palindromes consisting of the letters d, e, n, o, r, and v. The palindrome may contain spaces and periods which should be filtered out by the lexer. Define the tokens you use in your grammar using regular expressions. Define a regular expression which matches characters to be filtered.

3. Show how solution would accept:

   never odd or even.

   by first showing the tokenization of the string and then showing the complete parse tree generated.
Problem 2:
Build a Deterministic Finite Automaton (DFA) that recognizes the following regular expression:

\[(ab|c)^*ba\]

Problem 3:
Is the following grammar in SLR? Prove your answer in an organized manner.

\[
\begin{align*}
S' & \rightarrow S$ \\
S & \rightarrow bA c \\
S & \rightarrow B c \\
A & \rightarrow d \\
A & \rightarrow A a B \\
B & \rightarrow d
\end{align*}
\]

Problem 4:
TRUE/FALSE:

1. If \( r \) is a regular expression, then \((r+)^* = r^* \)
2. If \( r \) is a regular expression, then \((r^*)+ = r^* \)
3. Every NFA can be algorithmically transformed into an equivalent DFA
4. YACC is a SLR parser generator
5. Changing the order of grammar rules in YACC does not affect conflict resolution
6. Static links are only necessary for languages that allow nested function declarations
7. Register accesses are faster than memory
8. Stacks are FIFO (First-In-First-Out) structures
9. In a machine where parameters are passed through statically allocated blocks of memory, recursion is not possible
Extra Credit:
The names of some programming languages are palindromes. List as many as you can. Extra credit will be given for each one listed.