Consider the following grammar G:

\[
\begin{align*}
S & \rightarrow A S | \epsilon \\
A & \rightarrow a A | b
\end{align*}
\]

1 [40]. If G is a grammar of the following type, give the appropriate parsing tables; if not, explain why not. (Note: Remember to give parsing tables if the grammar is of the indicated type. A yes/no answer is not sufficient.)

   (a) LL(1)
   (b) LR(0)
   (c) SLR(1)
   (d) LR(1)
   (e) LALR(1)

2 [10]. Show that the language generated by G is a regular language.

3 [10]. What are the characteristics of an SLR(0) grammar? In particular, compare SLR(0) grammars to one or more of the classes of grammars that we have already studied (e.g., the 5 classes asked for in question 1 above). Explain your answer.

4 [8]. Give the prefix and postfix for each of the following:

   (a) \((a+b)/(c+d)+(e+f)\)
   (b) \(a/b/c+d+e\)

5 [10]. Consider the regular expression: \((0^*1)^*\)

Give the minimal state DFA that accepts this set.

6 [12]. Attribute grammars:

   (a) In YACC, rules like:

   \[
   \text{Expr} : \text{Expr} \ '+' \text{Term} \ [\$\$ = 1 + \$3] \]

   represent (inherited, synthesized) [pick one] attributes.

   (b) Why does YACC use an LR or LALR parsing algorithm rather than an LL parsing algorithm? Explain this relative to the question of using attribute grammars.

   (c) Could a tool somewhat like YACC be developed that used a precedence parsing algorithm? Explain.

   (d) Assume you have an attribute \texttt{code} that output the postfix for a nonterminal, and an attribute \texttt{type} that gave the type of data represented by a nonterminal (either \texttt{integer} or \texttt{real}). Give a possible definition for \texttt{Expr\ code} in the production of part (a) that allows for coercion of integer to real data.

7 [10]. Answer or explain each of the following statements with a short explanation (2-3 sentences each).

   (a) The pumping lemma can be used to show that a given language can be generated by a context free grammar.

   (b) If a grammar is ambiguous, the language it generates cannot be LR(1).

   (c) Since the regular expression \(a^*b^*\) can be used to generate \(a^n b^n\), that means the language \(a^n b^n\) is a regular language since it is generated by a regular expression.