1. (a) Finite automata (both DFAs and NFAs), and therefore regular expressions, have a finite memory of the characters read. Recognizing plaindromes of arbitrary length requires infinite memory.

   (b) The following grammar will generate arbitrary length palindromes over the given alphabet:

   | | | | | | |
   |---|---|---|---|---|---|
   | S→dSd | S→eSe | S→nSn | S→oSo | S→rSr | S→vSv |
   | S→d | S→e | S→n | S→o | S→r | S→v |
   | S→ $\epsilon$ | | | | | |

   To recognize the language we could use the tokens:

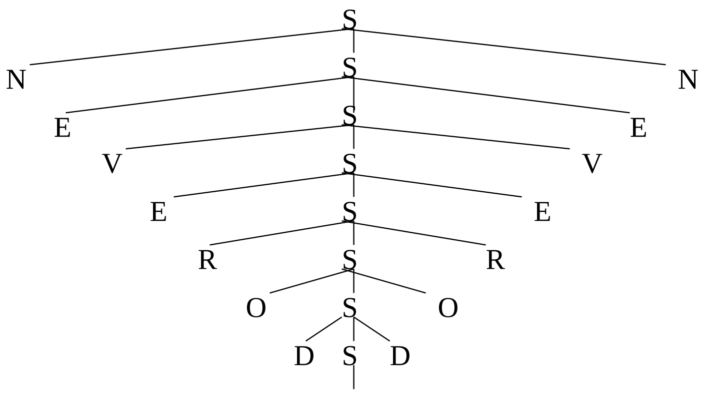
   **D E N O R V**

   With the lexer using the regular expressions (The last one will filter out spaces and periods):

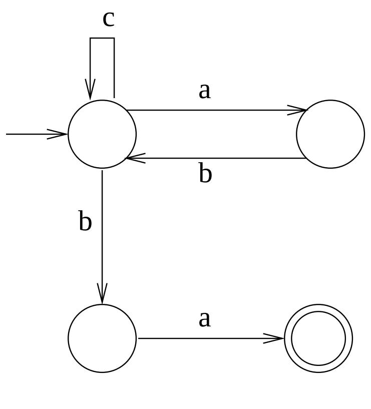
   d⇒**D** e⇒**E** n⇒**N** o⇒**O** r⇒**R** v⇒**V**

   [ .]$^+$ ⇒ continue()

   (c) The following string of tokens will be recognized: **N E V E R O D D O R E V E N**. The parse tree looks like:

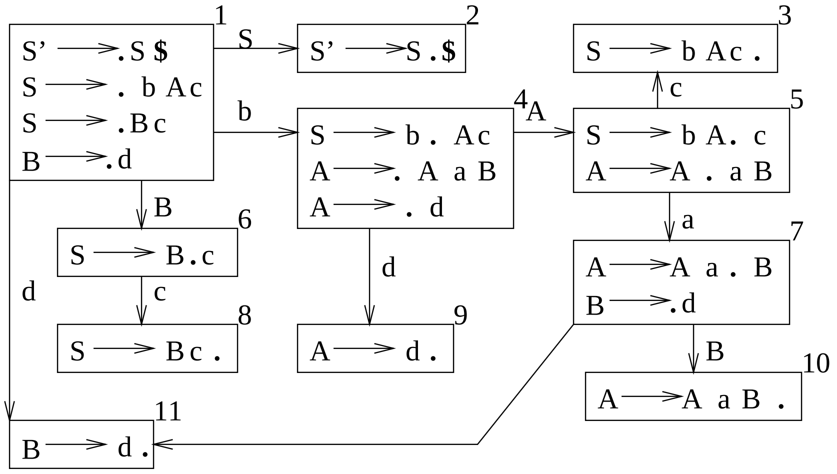


2. The simplest DFA for the expression `(ab|c)*ba` is:



All possible arrows not shown are assumed to lead into a state which has a transition to itself on all possible inputs (a reject state). There are other variants which are correct.

3. The grammar is SLR. In fact the grammar is LR(0). The state diagram is:



with the corresponding parse table:

| State | **a** | **b** | **c** | **d** | **$** | S | A | B |
|---|---|---|---|---|---|---|---|---|
| 1 | | s4 | | s11 | | g2 | | g6 |
| 2 | | | | | a | | | |
| 3 | r2 | r2 | r2 | r2 | r2 | | | |
| 4 | | | | s9 | | | g4 | |
| 5 | s7 | | s3 | | | | | |
| 6 | | | s8 | | | | | |
| 7 | | | | s11 | | | | g10 |
| 8 | r3 | r3 | r3 | r3 | r3 | | | |
| 9 | r4 | r4 | r4 | r4 | r4 | | | |
| 10 | r5 | r5 | r5 | r5 | r5 | | | |
| 11 | r6 | r6 | r6 | r6 | r6 | | | |

The table has no conflicts (i.e. there is no state where both a shift and a reduce action are possible), so the grammar is LR(0). Since all LR(0) grammars are SLR, this grammar is SLR.

4. (a) True
   (b) True
   (c) True
   (d) False
   (e) False

(f) True

(g) True

(h) False

(i) True