1. Consider the following BNF definition of boolean expressions in an imaginary programming language:

\[
\langle \text{bool-expr} \rangle \rightarrow \text{true} \mid \text{false} \mid \text{id} \mid \text{not} \langle \text{bool-expr} \rangle \mid \langle \text{bool-expr} \rangle \text{ and } \langle \text{bool-expr} \rangle \\
\mid \langle \text{bool-expr} \rangle \text{ or } \langle \text{bool-expr} \rangle \mid \langle \text{bool-expr} \rangle\'\langle \text{bool-expr} \rangle\'\'
\]

The single quotes around ‘(‘ mean that ( is a token in this definition, and similarly for ‘)’.

(a) (3 pts) Give a leftmost derivation for the string: (false or id) and id
(b) (3 pts) Draw a parse tree for the string: (false or id) and id
(c) (6 pts) Show that this BNF definition is ambiguous.

2. (4 pts) The following EBNF description is for all strings of 0’s and 1’s such that each 0 is followed by at least one 1.

\[
\langle \text{zero-one} \rangle \rightarrow \{(01 \mid 1)\}
\]

Give an equivalent BNF definition – that is, a definition of the same strings of 0’s and 1’s without using any of the extensions from EBNF.

3. (4 pts) Write a BNF (not EBNF) definition that describes all strings of 1’s that are of even length. Hint: 0 is an even number.

4. (4 pts) Describe (in English) the language generated by the following BNF definition. (Recall that the language generated by a BNF definition is the set of all strings of tokens that can be derived from the start symbol of the grammar.)

\[
\langle \text{S} \rangle \rightarrow \langle \text{A} \rangle \langle \text{B} \rangle \langle \text{C} \rangle \\
\langle \text{A} \rangle \rightarrow \text{a} \langle \text{A} \rangle \mid \text{a} \\
\langle \text{B} \rangle \rightarrow \text{b} \langle \text{B} \rangle \mid \text{b} \\
\langle \text{C} \rangle \rightarrow \text{c} \langle \text{C} \rangle \mid \text{c}
\]