Components of syntax

1. True or false: Punctuation such as semi-colons and commas is ignored in a syntax definition.
   - True
   - False

Which of the following are the lexemes in the code below?

```
SUM = 0.0
N = 1
WHILE N < 10
READ ITEM
SUM = SUM + ITEM
N = N + 1 WEND
PRINT SUM
```

1. WEND
2. N + 1
3. ITEM
4. PRINT SUM

- 1 and 2
- 2 and 3
- 1 and 3
- 3 and 4

3. Are reserved words tokens?
   - Yes
   - No

4. How many tokens does the following code fragment have?

```
while (x >= 0) {
    if (x == 3) break;
    x = analyze();
}
```

- 19
- 20
- 22

5. True or false. Whitespace characters, such as space, tab and linefeed can be safely ignored in computer languages.
   - True
   - False
Types of grammar, BNF.

1. True or false. A token is called a non-terminal symbol in BNF.
   - True
   - False

2. True or false. Terminal symbols are surrounded by diamond brackets (<>).
   - True
   - False

3. True or false. The order of elements on the right-hand side of a rule is unimportant.
   - True
   - False

4. True or false. BNF allows only one non-terminal on the left-hand side of a rule.
   - True
   - False

5. True or false. Recursion between rules is allowed in BNF. e.g.
   \[
   \texttt{<nt1>} ::= \ldots \texttt{<nt2>} \ldots \\
   \texttt{<nt2>} ::= \ldots \texttt{<nt1>} \ldots
   \]
   - True
   - False
Parse trees.

1. Given the grammar:
   
   \[
   \langle e \rangle ::= \langle b \rangle \mid \langle b \rangle \langle e \rangle \\
   \langle b \rangle ::= [ \langle e \rangle ] \mid []
   \]

   which of the following trees show the correct structure for the sentence \[ [ [ ] ] [ ] \]?

   ![Diagram of trees]

   - 1
   - 2

2. Given the grammar rule:
   
   \[
   \langle expr \rangle ::= \langle num \rangle \mid \langle expr \rangle + \langle expr \rangle
   \]

   Which of the following are correct derivations for \( \langle expr \rangle + \langle num \rangle \) ?
   
   - \( \langle num \rangle + \langle expr \rangle + \langle num \rangle \)
   - \( \langle expr \rangle + \langle expr \rangle + \langle num \rangle \)
   - \( \langle num \rangle + \langle num \rangle + \langle num \rangle \)
   - \( \langle expr \rangle + \langle expr \rangle + \langle expr \rangle \)

3. True or false. A parse tree can be infinitely big for a given finite sentence.
   
   - True
   - False

4. True or false. There is only one possible grammar for integers with given finite range of values.
   
   - True
False

True or false. A parse tree can only have one root node.
- True
- False
Ambiguity in grammars.

1. True or false. Does the ambiguity in the following rule matter semantically?
   \[ \text{<expr> } \text{<num>} \mid \text{<expr>} + \text{<expr>} \]
   - True
   - False

2. True or false. C allows expressions like \( x = y = 0 \) where the assignment operator is right associative. Is the following grammar ambiguous?
   \[ \text{<expr>} ::= \text{<assign>} \mid \text{id} \mid \ldots \]
   \[ \text{<assign>} ::= \text{id} = \text{<expr>} \]
   - True
   - False
Syntax: EBNF

1. Which of the following EBNF forms removes the recursion in
   \[ \text{par-list} ::= \text{par} | \text{par}, \text{par-list} \]
   - \[ \text{par-list} = (',', \text{par}) \]
   - \[ \text{par-list} = \text{par} (\text{par},')' \]
   - \[ \text{par-list} = \text{par},' (\text{par}) \]
   - \[ \text{par-list} = \text{par} (',', \text{par}) \]

2. Which of the following EBNF forms can replace the alternatives in
   \[ \text{end-tag} ::= \text{end} | \text{end} \text{id} \]
   - \[ \text{end-tag} = ['end' \text{id}] \]
   - \[ \text{end-tag} = ['end'] \text{id} \]
   - \[ \text{end-tag} = 'end' [\text{id}] \]
   - \[ \text{end-tag} = 'end' ['end' \text{id}] \]

3. True or false. Can the repetition in the following EBNF rule be replaced by recursion in a single BNF rule?
   \[ \text{block} = 'begin' \{\text{stmt} ';'}\} \text{stmt} 'end' \]
   - True
   - False
Syntax diagrams

1. Which EBNF rule is the equivalent of the diagram

```
float = ('0' | {digit}+) '.' {digit}+
float = '0' | {digit}+ '.' {digit}+
float = ('0' {digit}+) '.' {digit}+
float = ('0' | {digit}) '.' {digit}
```

2. True or false. Are the following diagrams equivalent?

```
stmt-list
```

```
stmt-list
```

1. True
2. False