Q1: Parse Tree and Leftmost Derivation for Four Statements

The BNF grammar:

```
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <expr> + <term> | <term>
<term> → <term> * <factor> | <factor>
<factor> → ( <expr> ) | <id>
```

1. A = (A + B) * C

Leftmost derivation:

```
<assign>
<id> = <expr>
A = \langle expr \rangle
A = < term >
A = <term> * <factor>
A = <factor> * <factor>
A = ( < expr > ) * < factor >
A = ( <expr> + <term> ) * <factor>
A = ( <term> + <term> ) * <factor>
A = ( < factor > + < term > ) * < factor >
A = ( <id> + <term> ) * <factor>
A = (A + < term >) * < factor >
A = (A + < factor>) * < factor>
A = (A + <id>) * <factor>
A = (A + B) * < factor>
A = (A + B) * <id>
A = (A + B) * C
```

Parse tree:

2.
$$A = B + C + A$$

Leftmost derivation:

Parse tree:

3.
$$A = A * (B + C)$$

Leftmost derivation:

```
<assign>
<id> = <expr>
A = \langle expr \rangle
A = < term >
A = < term > * < factor >
A = <factor> * <factor>
A = <id> * <factor>
A = A * < factor>
A = A * ( <expr> )
A = A * ( <expr> + <term> )
A = A * ( <term> + <term> )
A = A * ( < factor > + < term > )
A = A * ( <id> + <term> )
A = A * (B + < term>)
A = A * (B + < factor>)
A = A * (B + < id>)
A = A * (B + C)
```

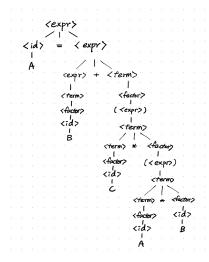
Parse tree:

4.
$$A = B + (C * (A * B))$$

Leftmost derivation:

```
<assign>
<id> = <expr>
A = \langle expr \rangle
A = \langle expr \rangle + \langle term \rangle
A = < term > + < term >
A = <factor> + <term>
A = \langle id \rangle + \langle term \rangle
A = B + < term >
A = B + < factor>
A = B + ( <expr> )
A = B + ( <term> * <factor> )
A = B + ( < factor > * < factor > )
A = B + ( <id> * <factor> )
A = B + (C * < factor>)
A = B + (C * (<expr>))
A = B + (C * (<term> * <factor> ))
A = B + (C * (< factor> * < factor> ))
A = B + (C * (<id> * <factor> ))
A = B + (C * (A * < id>))
A = B + (C * (A * B))
```

Parse tree:



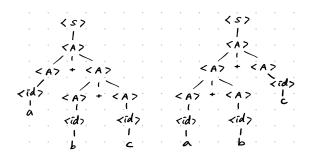
Q2: Prove Grammar is Ambiguous

The grammar is:

$$~~\rightarrow~~$$

 $\rightarrow + |$
 $\rightarrow a | b | c$

the string a + b + c can lead to 2 different parse trees:



If the structures of two parse trees are different, it indicate that the grammar is ambiguous.

Q3: Modify the Grammar to Add Unary Minus and Power Operator

- Precedence order: $'()' > > ^ > * > +$.
- Left associativity for + and * .
- Right associativity for and ^ .

Modified grammar:

```
<assign> \rightarrow <id> = <expr> <id> \rightarrow A | B | C <expr> \rightarrow <expr> \rightarrow <expr> + <term> | <term>
```

```
<term> → <term> * <factor> | <factor> <factor> → <unary> ^ <factor> | <unary> <unary> → - <primary> | <primary> → (<expr> ) | <id>
```