Assign 6:

1)

Example: let x - 5 in x; y

```
            <expr>
             / | \
        let   |  <expr>
           /   |    | \
         <id>  -   <expr>
          |          / | \
          x         x  ;  y
```

```
            <expr>
             / | \
        <expr> ; <expr>
         / | \      | \
      <expr> | <expr> | y
       / | \        \
  let   |  <expr>     x
     /   |    |
   <id>  -   <expr>
    |          / | \
    x         x  ;  y
```

In this parse tree, the semicolon (;) is treated as a separate expression.

The same input "let x - 5 in x; y" can be parsed in two different ways, resulting in different parse trees. This demonstrates the ambiguity in the given grammar.

2)

```
<id>    ::= a | b | c | ... | z
<dig>   ::= 0 | 1 | 2 | ... | 9
<expr>  ::= <term> | <term> ; <expr>
<term>  ::= <factor> | let <id> = <expr> in <term>
<factor>::= () | <dig> | <id> | begin <expr> end
```

3)

```
           <term>
             / | \
          let  | <id> = <expr> in <term>
             |        |         |
             x        5         x; y
```

Now, the let expression is modified to use <term> instead of <expr> for the body of the let. This makes it unambiguous by explicitly stating that the let expression can contain multiple terms separated by semicolons.In this parse tree, the let expression is clearly structured, and the semicolon (;) is associated with the subsequent term. This modification removes the ambiguity in the original grammar.