Week 2 Question:

There are two restrictions on the type of grammars that can be used with a recursive descent parser. The first is that the grammar cannot have any left recursive productions. Give an example of a left recursive production and explain why such productions would be a problem.

The second restriction is that the grammar must not require more than one token look ahead. Give an example of a production that does not have this property. Explain why this restriction is necessary for recursive descent parsing.

**There are two restrictions on the type of grammars that can be used with a recursive descent parser. The first is that the grammar cannot have any left recursive productions. Give an example of a left recursive production and explain why such productions would be a problem.**

Here’s some grammar I made up:

A -> B | C | D

B -> a | b

C -> A | B

D -> c | d

This grammar contains an indirect left recursion.  A’s FIRSTS are {a, b}, {A, B}, and {c, d}.  Because A is listed, there is a left recursion occurring.

When writing code to parse this grammar, the recursive method for A could call the method for C, which could then immediately call A again.  This would result in an indefinite loop.

**The second restriction is that the grammar must not require more than one token look ahead. Give an example of a production that does not have this property. Explain why this restriction is necessary for recursive descent parsing.**

Here is another grammar:

A- > a | b | B

B -> a | c | C

C -> d | e

Here, when looking at A, our possible FIRSTS are {a, b} and {a, c, d, e}.  Because *a* is listed in both possibilities, it is impossible to determine which right hand side the *a* belongs to.  Because the parser only looks one step ahead, it would be unable to proceed without making an assumption about which right hand side to chose.