**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.**

Grammar:

X -> y | Z | D

Y -> X | b | e

Z -> c | Y | f

a -> c | d | g

The grammar has a buried indirect left recursion (in an effort to be different from others I buried it) the chances of the loop happening are relatively improbable, we must have NO loops like this. The loop is X calls Z, Z calls Y, and Y calls X.

Infinite loops are never desired, and even if its not an “infinite loop” per se like my example. We are still waiting compute resources doing loops with only a chance of getting out of it. This equates to wasted cycles.

**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.**

X- > x | y | Y

Y -> x | z | Z

Z -> a | b | c

In this example the grammar would need to look past one token to look ahead enough to know which right hand side to choose, allowing the grammar to continue. You cannot have to choices of x

(X -> x, Y -> x)