Lab Three

Alexis Dionne

Alexis.Dionne1@Marist.edu

February 27, 2019

Crafting a Compiler

4.7.a

<Start>
<E><$>
<T> plus <E><$>
<F> plus <E><$>
num plus <E><$>
num plus <T><$>
num plus <T> times <F><$>
num plus <F> times <F><$>
num plus num times <F><$>
num plus num times (<E>) <$>
num plus num times (<T> plus <E>) <$>
num plus num times (<F> plus <E>) <$>
num plus num times (num plus <E>) <$>
num plus num times (num plus <T>) <$>
num plus num times (num plus <F>) <$>
num plus num times (num plus num) <$>
num plus num times (num plus num) $

4.7.B

<Start>
<E><$>
<E> $
<T> $
<T> times <F> $
<T> times num $
<F> times num $
(<E>) times num $
(<T> plus <E>) times num $
(<T> plus <T>) times num $
(<T> plus <F>) times num $
(<T> plus num) times num $
(<T> plus num) times num $
(<F> plus num) times num $
((<E>) plus num) times num $
((<T>) plus num) times num $
((<T> times <F>) plus num) times num $
((<T> times num) plus num) times num $
((<F> times num) plus num) times num $
((num times num) plus num) times num $

4.7.C

This grammar uses the opposite of PEMDAS to go in opposite operational precedence, and it also favors left-associativity of operators. I say both these things because plus is higher up in the rules of the grammer, and the derivation looks nicer and took fewer steps when done left to right.

5.2C

```
Parser(tokens)
    self.tokens = tokens
    self.currentToken = none

    parseStart()
        addBranchNode("start")
        parseValue()
        addLeafNode("$")

    parseValue()
        addBranchNode("value")
        if currentToken is num:
            matchAndConsume({num})
        else:
            matchAndConsume({ ( })
            parseExpr()
            matchAndConsume({ ) })
        returnToParent()
```

```
parseExpr()
    addBranchNode("expr")
    if currentToken is plus:
        matchAndConsume({plus})
        parseValue()
        parseValue()
    else:
        matchAndConsume({prod})
        parseValues()
    returnToParent()

parseValues()
    addBranchNode("values")
    if currentToken is not none:
        parseValue()
        parseValues()
    else:
        # do nothing bc lambda
    returnToParent()
```

Dragon

4.2.8

S -> S S + | S S * | a

string of a a + a *

A

```
<S>
<S><S>*
<S><S>+<S>*
a <S>+<S>*
a a + <S>*
a a + a *
```

B

```
<S>
<S><S>*
<S> a *
<S><S>+ a *
<S> a + a *
a a + a *
```

C

tree.jpg