Lab 3

Crafting a Compiler

4.7a and 4.7b:

Leftmost Derivation	Rightmost Derivation
E $	E $
T plus E $	T plus E $
F plus E $	T plus T $
num plus E $	T plus T times F $
num plus T plus E $	T plus T times num $
num plus T times F plus E $	T plus F times num $
num plus F times F plus E $	t plus num times num $
num plus num times num plus E $	$ times F plus num times num $
num plus num times num plus T $	T times num plus num times num $
num plus num times num plus F $	F times num plus num times num $
num plus num times num plus num $	num times num plus num times num $

4.7c. In this grammar, expressions exist next to plus or times operations and are also composed of expressions with the plus or times operators, with ‘num’ being acted on.

5.2c Construct a recursive-descent parser based on the grammar.

start: parseValue()

parseValue: match(num) or match(lparen) -> parseExpr() -> match(rparen)

parseExpr: match(plus) -> parseValue() -> parseValue() or match(prod) -> parseValues()

parseValues(): parseValue() -> parseValues() or do nothing

match: check if current token matches the argument; if it does, move to next token.

Dragon Book

a. Leftmost derivation

S

SS

SS+

SS+S*

aS+S*

aa+S*

aa+a*

b. Rightmost derivation

S

SS*

Sa*

SS+a*

Sa+a*

aa+a*

c. Parse Tree

```
       S
     / | \
    S   S  *
 / | \  |
S  S  + a
|  |
a  a
```