Homework 2

Haden Stuart – has0027

1) (10) Given the grammar below, identify which sentences are in the language (which are valid sentence).
    a. baab
    b. bbbab
    c. bbaaaaaa
    d. bbaab

<S> → <A> a <B> b
<A> → <A> b | b
<B> → a <B> | a

From the first line, <S> -> <A> a <B> b, we know that the ending to any sentence must have the letter b, meaning c is invalid. Continuing through the grammar we have <A> -> <A> b | b, which means that the first letter must be the letter b. Plugging these values in we get two possible strings:

ba<B>b or bba<B>b ➔ baab or bbaab ➔ Meaning only **a** and **d** are valid

2) (10) Identify all of the tokens (categories of lexemes) in the grammar below, and which lexemes they categorize. Put them in a table.

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <id> + <expr>

| <id> * <expr>

| (<expr>)

| <id>

Tokens	Lexemes
Variable	A,B,C
Add_op	+
Mult_op	*
Assign_op	=
Parenthesis	(,)

3) (10) Given the grammar from question 2, show a left-most derivation and draw the parse tree for the following statement.

a. B = B + (C + (A * A))

<assign> -> <id> = <expr>

-> B = <expr>

-> B = <id> + <expr>

-> B = B + <expr>

-> B = B + (<expr>)

-> B = B + (<id> + <expr>)

-> B = B + (C + <expr>)

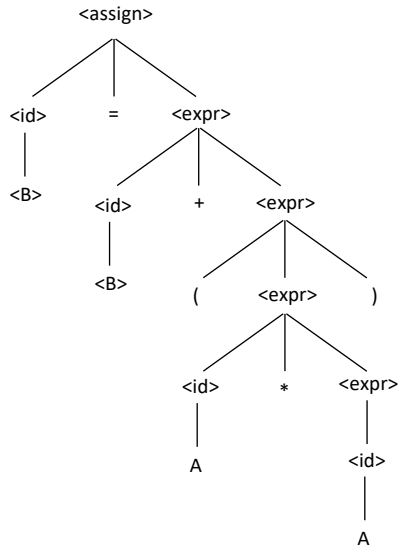
-> B = B + (C + (<expr>))

-> B = B + (C + (<id> * <expr>))

-> B = B + (C + (A * <expr>))

-> B = B + (C + (A * <id>))

-> B = B + (C + (A * A))



4) (10) Remove all of the recursion from the following grammar:

S -> Aa | Bb
A -> Aa | AbC | C
B -> S | bb
C -> c

S -> Aa | Bb
A -> AA'
A -> C
A' -> a | bC
B -> S | bb
C -> c

5) (10) Use left factoring to resolve the pairwise disjointness problems in the following grammar:

A -> aBc | ac | a
B -> b | aB

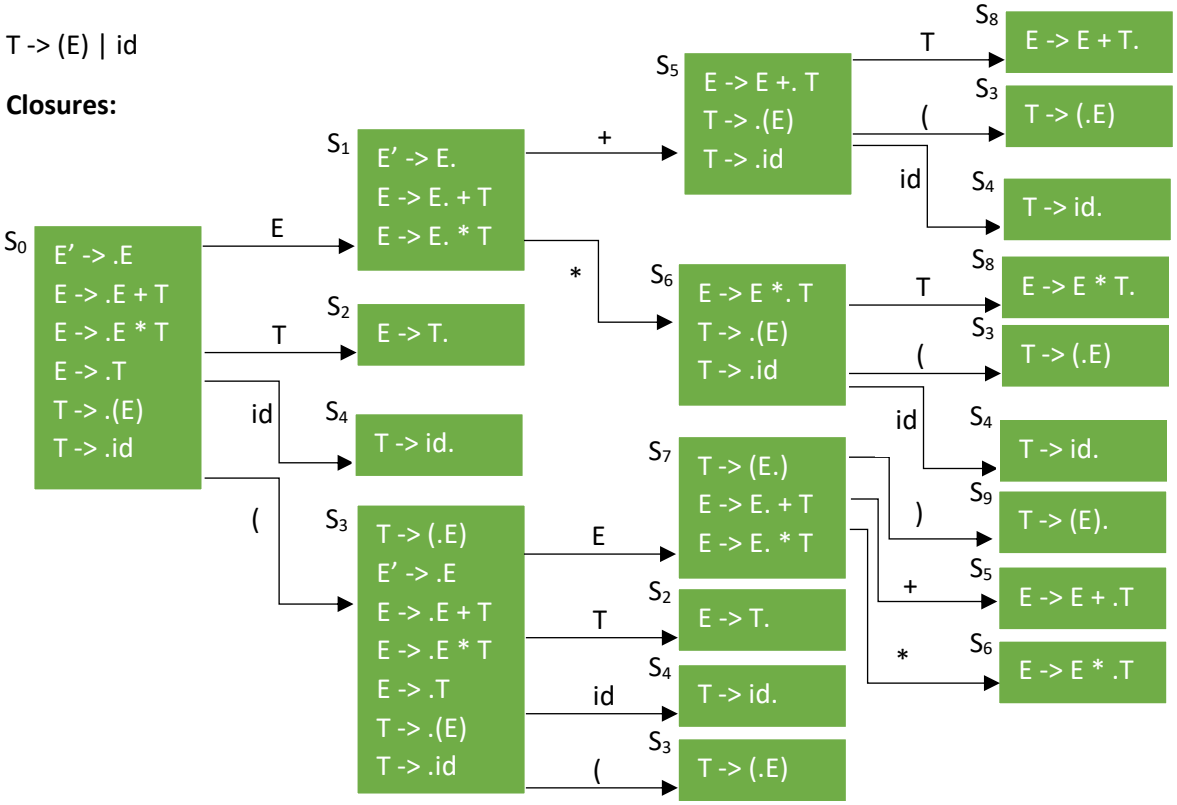
A -> aA'
A' -> Bc | c
B -> b | aB

6) (20 pts) Create an LR(0) parse table for the following grammar. Show all steps (creating closures, the DFA, the transition table, and finally the parse table):

E -> E + T | E * T | T

T -> (E) | id

Closures:



State	Action						Goto	
	+	*	(	)	Id	$	E	T
0			S3		S4	acc	1	2
1	S5	S6						
2	R3	R3		R3				
3			S3		S4		7	2
4	R5	R5		R5				
5			S3		S4			8
6			S3		S4			8
7	S5	S6		S9				
8	R2	R2		R2				
9	R4	R4		R4				

7) (20 pts) Show a complete bottom-up parse, including the parse stack contents, input string, and action for the string below using the parse table you created in step 6. Think about how I went through this in class.

Parse stack:

Input String:

(id + id) * id

8) (10 pts) Show a rightmost derivation for the string above, and show how the bottom-up parse you completed in step 7 correctly finds all of the handles for the input string above.

(id + id) * id