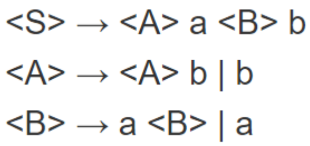
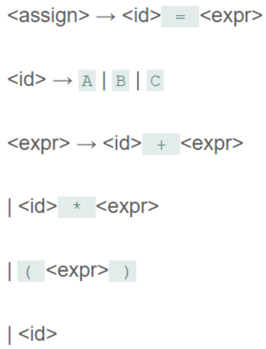
1) (10) Given the grammar below, identify which sentences are in the language (which are valid sentence).
   a. baab – This IS a valid sentence.
   b. bbbab – This is NOT a valid sentence.
   c. bbaaaaaa – This is NOT a valid sentence.
   d. bbaab – This is a valid sentence.



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

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
<id>	A, B, C
<eq_op>	=
<add_op>	+
<mult_op>	*
<left_paren>	(
<right_paren>	)

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>))

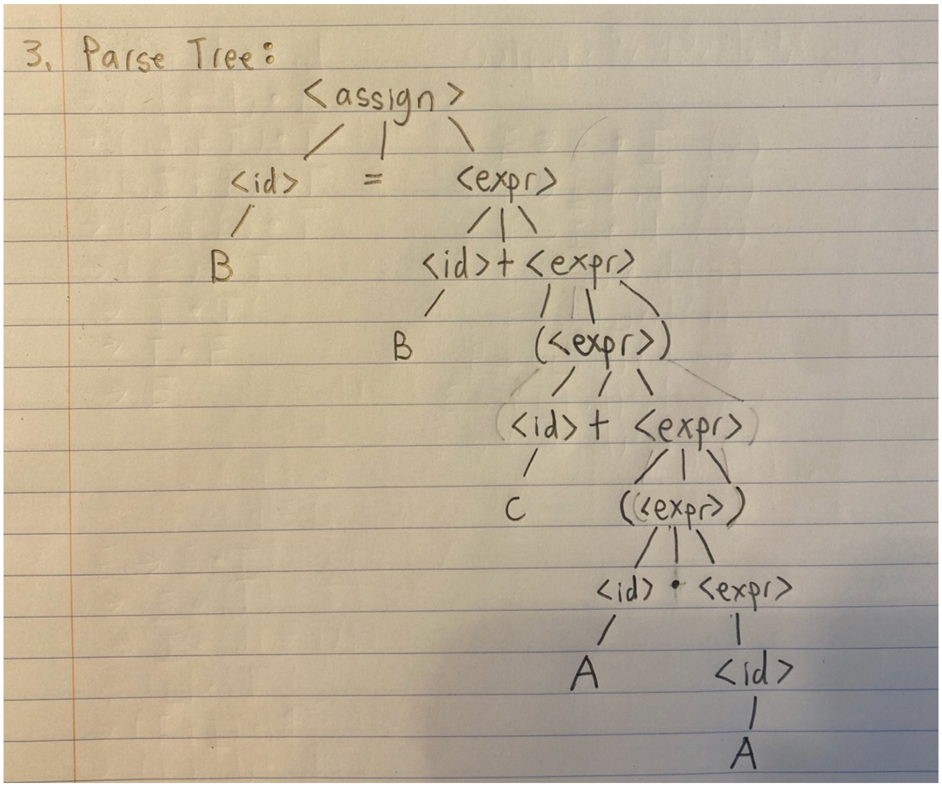
B = B + (C + (<id> * <expr>))

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

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

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

PARSE TREE:



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

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

S → AaS' | bbbS'

S' → bS' | epsilon

A → CA'

A' → aA' | bCA' | epsilon

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 → aC

B → b | aB

C → Bc | c | epsilon

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

RULES:

R0: S' → E$

R1: E → E + T

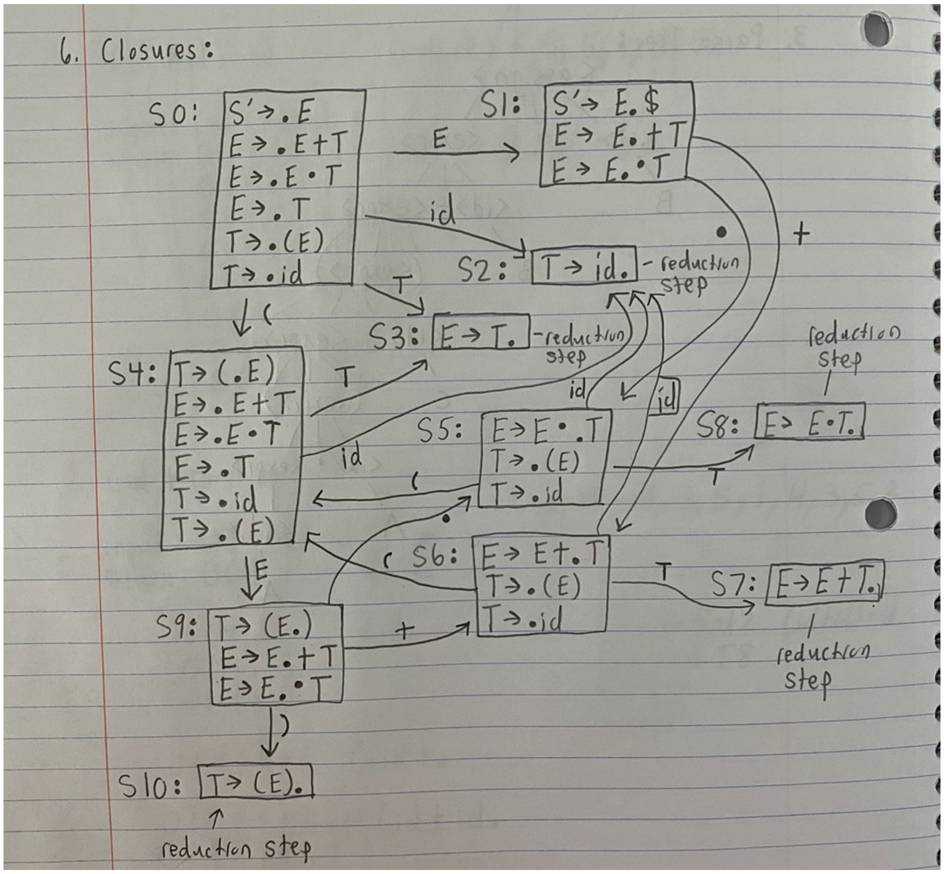
R2: E → E * T

R3: E → T

R4: T → (E)

R5: T → id

CLOSURES:



TRANSITION TABLE:

	E	T	id	*	+	(	)
0	1	3	2			4	
1				5	6		
2							
3							
4	9	3	2				
5		8	2			4	
6		7	2			4	
7							
8							
9				5	6		10
10							

PARSE TABLE:

	ACTION						GOTO	
	id	*	+	(	)	$	E	T
0	S2			S4			1	3
1		S5	S6			acc		
2	R5	R5	R5	R5	R5			
3	R3	R3	R3	R3	R3			
4	S2			S4			9	3
5	S2			S4				8
6	S2			S4				7
7	R1	R1	R1	R1	R1			
8	R2	R2	R2	R2	R2			
9		S5	S6		S10			
10	R4	R4	R4	R4	R4			

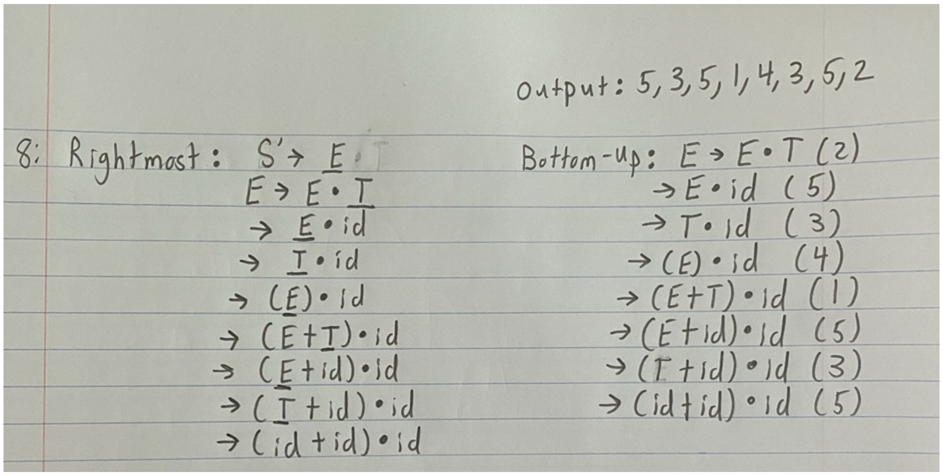
7) (20 pts) Show a complete bottom-up parse, including the parse stack contents, input string, and action for the string below using the parse table you created in step 6. Think about how I went through this in class.

(id + id) * id

STACK	INPUT	ACTION
0	.(id + id) * id $	Shift 4
0 (4	(.id + id) * id $	Shift 2
0 (4 id 2	(id. + id) * id $	Reduce by T→id (R5)
0 (4 T 3	(id. + id) * id $	Reduce by E→T (R3)
0 (4 E 9	(id. + id) * id $	Shift 6
0 (4 E 9 + 6	(id +. id) * id $	Shift 2
0 (4 E 9 + 6 id 2	(id + id.) * id $	Reduce by T→id (R5)
0 (4 E 9 + 6 T	(id + id.) * id $	Reduce by E→E + T (R1)
0 (4 E 9	(id + id.) * id $	Shift 10
0 (4 E 9) 10	(id + id). * id $	Reduce by T→(E) (R4)
0 T 3	(id + id). * id $	Reduce by E→T (R3)
0 E 1	(id + id). * id $	Shift 5
0 E 1 * 5	(id + id) *. id $	Shift 2
0 E 1 * 5 id 2	(id + id) * id. $	Reduce by T→id (R5)
0 E 1 * 5 T 8	(id + id) * id. $	Reduce by E→E * T (R2)
0 E 1	(id + id) * id. $	Accept
0 E 1	(id + id) * id $.	---------

OUTPUT: 5, 3, 5, 1, 4, 3, 5, 2

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.



Output: 5, 3, 5, 1, 4, 3, 5, 2

8: Rightmost: $S' \to \underline{E}$

$E \to E \bullet \underline{T}$

$\to \underline{E} \bullet id$

$\to \underline{T} \bullet id$

$\to (\underline{E}) \bullet id$

$\to (E + \underline{T}) \bullet id$

$\to (\underline{E} + id) \bullet id$

$\to (\underline{T} + id) \bullet id$

$\to (id + id) \bullet id$

Bottom-up: $E \to E \bullet T$ (2)

$\to E \bullet id$ (5)

$\to T \bullet id$ (3)

$\to (E) \bullet id$ (4)

$\to (E + T) \bullet id$ (1)

$\to (E + id) \bullet id$ (5)

$\to (T + id) \bullet id$ (3)

$\to (id + id) \bullet id$ (5)