Compiler Construction – Practice Sheet.

Part A:- LL(1) Parsing Practice Problems:

Step For each Problem.

Step 1:- Compute FIRST sets.
Step 2:- Compute FOLLOW sets.
Step 3:- Construct the LL(1) Parsing Table.
Step 4:- Parse the Input String using Stack.
Step 5:- Construct the Parse Tree

Problem 1

$E \rightarrow TE'$
$E' \rightarrow +TE' | \varepsilon$ input = id + id * num
$T \rightarrow FT'$
$T' \rightarrow *FT' | \varepsilon$
$F \rightarrow (E) | id | num.$

Step 1:- ① First and Follow sets

E ⇔	First( )	Follow( )
$E \rightarrow TE'$	{(, id, num}	{$,)}
$E' \rightarrow +TE' \mid \varepsilon$	{+, ε}	{$,)}
$T \rightarrow FT'$	{(, id, num}	{+, $,)}
$T' \rightarrow *FT' \mid \varepsilon$	{*, ε}	{+, $,)}
$F \rightarrow (E) \mid id \mid num$	{(, id, num}	{*, +, $,)}

Step 2:- Parse Table

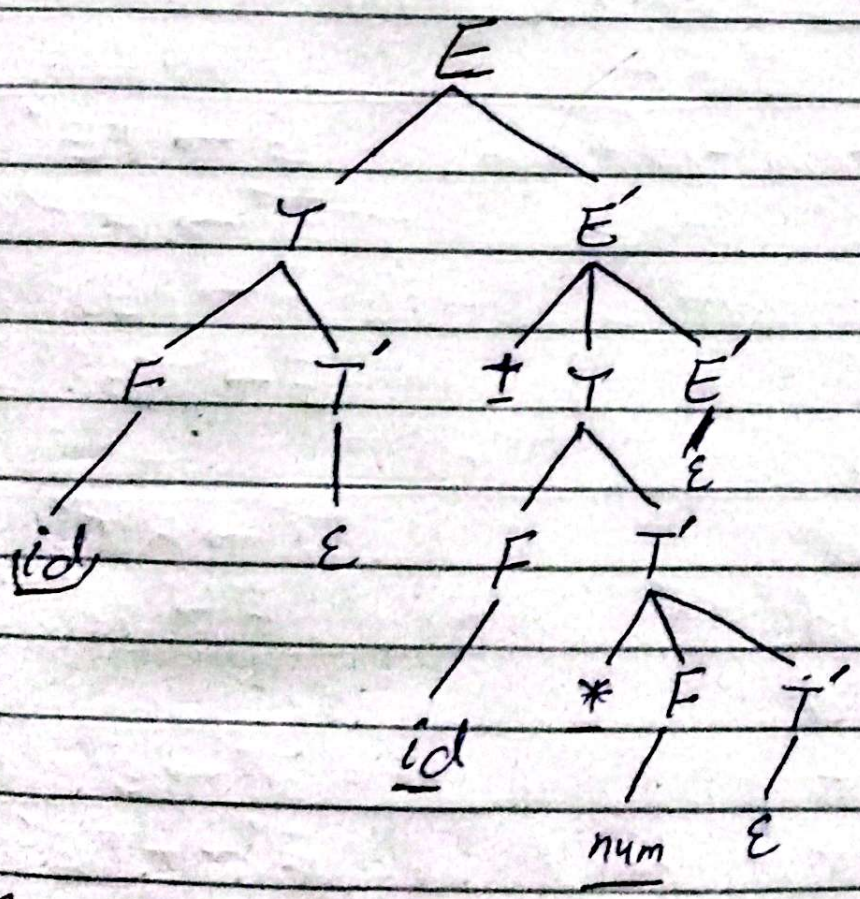
	(	)	id	num	+	*	$
E	$E \rightarrow TE'$		$E \rightarrow TE'$	$E \rightarrow TE'$			
E'		$E' \rightarrow \varepsilon$			$E' \rightarrow +TE'$		$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$		$T \rightarrow FT'$	$T \rightarrow FT'$			
T'		$T' \rightarrow \varepsilon$			$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$	$T' \rightarrow \varepsilon$
F	$F \rightarrow (E)$		$F \rightarrow id$	$F \rightarrow num$			

Step 3:- Input Buffer- | id | + | id | * | num | $ |

Stack	Input	Action
$ E	id.	$E \rightarrow TE'$
$ E'T	id	$T \rightarrow FT'$
$ E'T'F	id	$F \rightarrow id$
$ E'T' id ←	→ id (match, pop from stack.)	

Stack	Input	Action.
$ E T'	+ (next input)	T' → ε
$ E E' (no effect.)	+	E' → +TE'
$ E' T + ← match → +		
$ E' T	id (next input).	T → FT'
$ E' T' F	id.	F → id
$ E' T' id ← match → id		
$ E' T'	* (next input)	T' → *FT'
$ E' T' F * ← match → *		
$ E' T' F	num (next input)	F → num
$ E' T' num ← match → num		
$ E' T'	$	T' → ε
$ E' ε	$	E' → ε
$ ε	$	
$ ← match → $		String accept.

Step 4:



Problem 3:-

E → TE'

E' → +TE' | -TE' | ε

T → FT'

T' → *FT' | ε

F → id | (E)

Input:- (id + num) - id * num

Step 1:- First and Follow sets

	First()	Follow()
$E \to TE'$	{id, (, num}	{$,)}
$E' \to +TE' \mid -TE' \mid \varepsilon$	{+, -, ε}	{$,)}
$T \to FT' \varepsilon$	{id, (, num}	{+, -, $,)}
$T' \to *FT' \mid \varepsilon$	{*, ε}	{+, -, $,)}
$F \to id \mid (E) \mid num.$	{id, (, num}	{*, +, -, $,)}

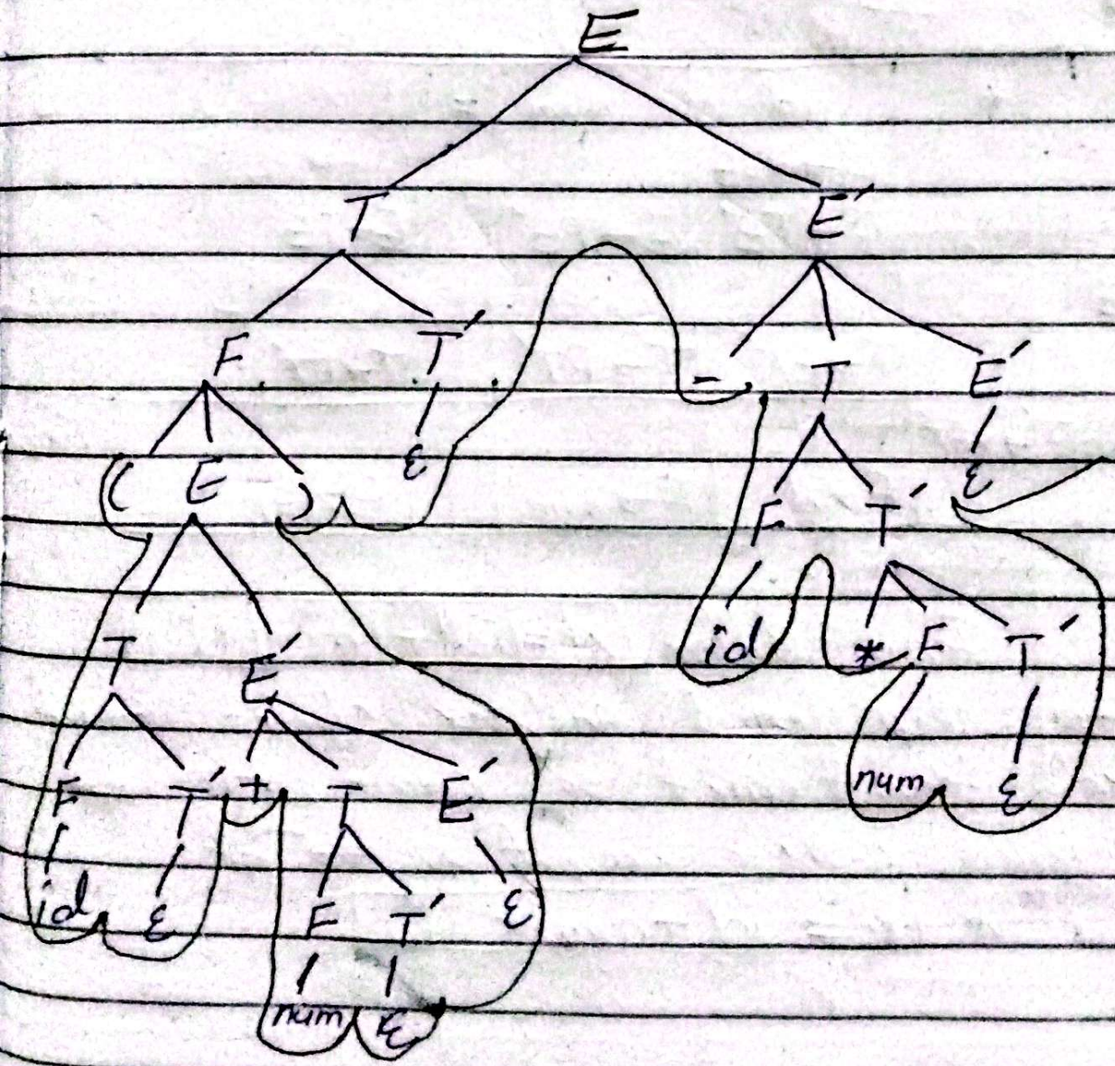
Step 2:- Parse Table

	+	-	*	(	)	id num	num	$
E				E→TE'		E→TE'	E→TE'	
E'	E'→+TE'	E'→-TE'			E'→ε			E→ε
T				T→FT'		T→FT'	T→FT'	
T'	T'→ε	T'→ε	T'→*FT'		T'→ε			T→ε
F				F→(E)		F→id	F→num	

Step 3:- Input Buffer. [(| id | + | num | - | id | * | num | $]

Stack	Input	Action.
$E	(id+num)-id*num	E → TE'
$E'T	(	T → FT'
$E'T'F	(	F → (E)
$E'T')E(	match → (	E → TE'
$E'T')E	id	
$E'T')ET	id	T → FT'
$E'T')ETF	id	F → id
$E'T')ET'id	match → id	
$E'T')ET'	+	T' → ε
$E'T')E'	+	E' → +TE'
$E'T')E'T+	match → +	
$E'T')E'T	num	T → FT'
$E'T')E'TF	num	F → num
$E'T')E'Tnum	match → num	
$E'T')E'T	)	T' → ε

Stack	Input	Action
$ E'T')E'	)	E' → ε
$ E'T') ← match →)		
$ E'T'	-	T' → ε
$ E'	-	E' → -TE'
$ E'T- ← match → -		
$ E'T	id	T → FT'
$ E'T'F	id	F → id
$ E'T'id ← match → id		
$ E'T'	*	T' → *FT'
$ E'T'F* ← match → *		
$ E'T'F	num	F → num
$ E'T'num ← match → num		
$ E'T'	$	T' → ε
$ E'	$	E' → ε
$ ← match → $		Accept String



Problem 4.

$S \rightarrow id = E$

$E \rightarrow TE'$

$E' \rightarrow +TE' | \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' | \varepsilon$

$F \rightarrow id | num | (E)$

input x = id + num * id.

Step 1: First and Follow sets

	First()	Follow()
$S \rightarrow id = E$	{id}	{$}
$E \rightarrow TE'$	{id, num, (}	{), $}
$E' \rightarrow +TE' \mid \varepsilon$	{+, ε}	{), $}
$T \rightarrow FT'$	{id, num, (}	{+,), $}
$T' \rightarrow *FT' \mid \varepsilon$	{*, ε}	{+,), $}
$F \rightarrow id \mid num \mid (E)$	{id, num, (}	{*, +,), $}

Step 2: Parse Table

	+	*	id	num	=	(	)	$
S			$S \rightarrow id = E$					
E			$E \rightarrow TE'$	$E \rightarrow TE'$		$E \rightarrow TE'$		
E'	$E' \rightarrow +TE'$							$E' \rightarrow \varepsilon$
T			$T \rightarrow FT'$	$T \rightarrow FT'$		$T \rightarrow FT'$		
T'	$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$						$T' \rightarrow \varepsilon$
F			$F \rightarrow id$	$F \rightarrow num$		$F \rightarrow (E)$		

Step 3:- Input Buffer | x | = | id | + | num | * | id |

In this step lexical analyser doesn't generate token correct so correct this apply lexical analyser convert x → id.

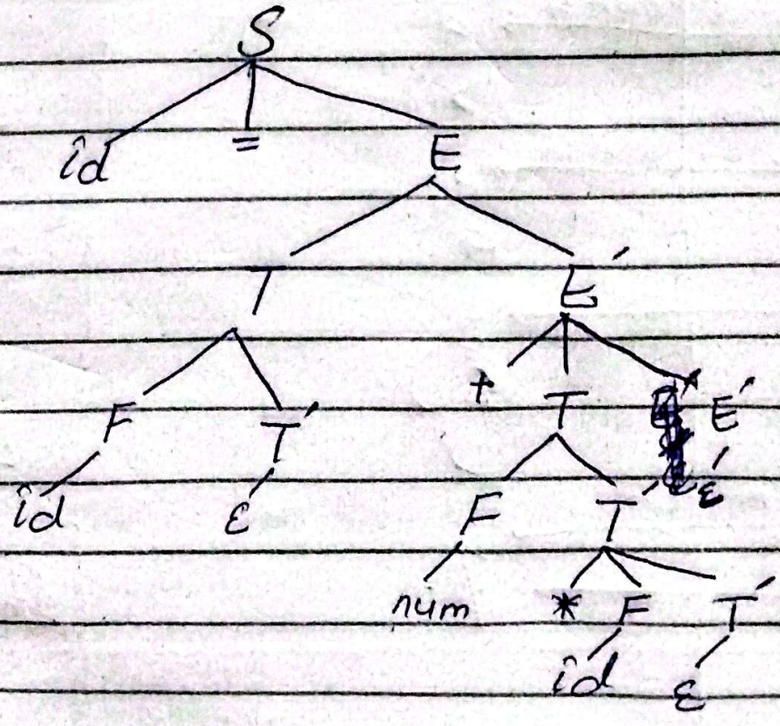
No input is [id = id + num * id.]

Input Buffer: | id | = | id | + | num | * | id | $ |

Stack	Input	Action
$S	id	S → id = E
$ E = id ← match → id		
$ E = ← match → =		
$ E	id	E → TE′
$ E′T	id	T → FT′
$ E′T′F	id	F → id
$ E′T′ id ← match → id		
$ E′T′	+	T′ → ε
$ E′	+	E′ → +TE′
$ E′T+ ← match → +		
$ E′T	num	T → FT′
$ E′T′F	num	F → num
$ E′T′ num ← match → num		
$ E′T′	*	T′ → *FT′
$ E′T′F* ← match → *		
$ E′T′F	id	F → id
$ E′T′ id ← match → id		
$ E′T′	$	T′ → ε
$ E′	$	E′ → ε
$ ← match → $		String accept.

Step 4:-

Parse Tree



Problem 2:

$S \to (L) \mid a$

$L \to SL'$

$L' \to , SL' \mid \varepsilon$

input: (a, a)

Step 1: First and Follow sets

	First()	Follow()
$S \to (L) \mid a$	{(, a}	{$, , ,)}
$L \to SL'$	{(, a}	{)}
$L' \to , SL' \mid \varepsilon$	{, , ε}	{)}

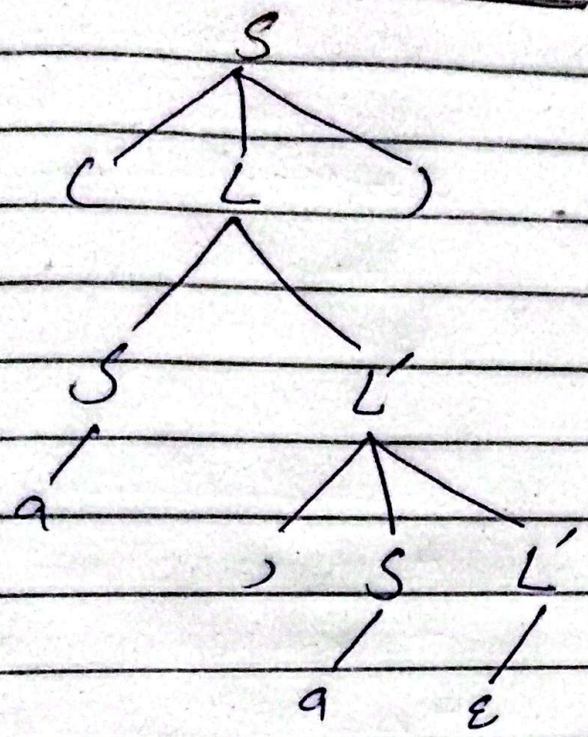
Step 2: Parse Table:

	,	(	)	a	$
S		$S \to (L)$		$S \to a$	
L		$L \to SL'$		$L \to SL'$	
L'	$L' \to , SL'$		$L' \to \varepsilon$		

Step 3: Input Buffer: | (| a | , | a |) | $ |

Stack	Input	Action
$S	(	$S \to (L)$
$)L(← match →	(	
$)L	a (next input)	$L \to SL'$
$)L'S	a	$S \to a$
$)L'a ← match →	a	
$)L'	,	$L' \to , SL'$
$)L'S, ← match →	,	
$)L'S	a	$S \to a$
$)L'a ← match →	a	
$)L'	)	$L' \to \varepsilon$
$)L'	)	
$) ← match →	)	
$ ← match →	$	String accept.
$		

Step 4:- Parse Tree.



Compiler Construction — Practice Sheet

Instructions

This practice sheet is divided into **two parts**:

- **Part A:** LL(1) Parsing Practice Problems (with given input strings)
- **Part B:** Context-Free Grammar (CFG) Construction Problems

---

Part A — LL(1) Parsing Practice Problems

For **each problem**, students must complete the following steps:
Step 1: Compute FIRST sets
Step 2: Compute FOLLOW sets
Step 3: Construct the LL(1) Parsing Table
Step 4: Parse the Input String using Stack
Step 5: Construct the Parse Tree

Problem 1

```
E  → T E'
E' → + T E' | ε
T  → F T'
T' → * F T' | ε
F  → ( E ) | id | num
```

Input: `id + id * num`

Problem 2

```
E  → T E'
E' → - T E' | ε
T  → F T'
T' → / F T' | ε
F  → ( E ) | id | num
```

Input: `id - id / num`

Problem 3

```
E  → T E'
E' → + T E' | - T E' | ε
T  → F T'
T' → * F T' | ε
F  → id | ( E ) | num
```

Input: `( id + num ) - id * num`

Problem 4

```
S  → id = E
E  → T E'
E' → + T E' | ε
T  → F T'
T' → * F T' | ε
F  → id | num | ( E )
```

Input: `x = id + num * id`

Problem 5

```
S  → id = E
E  → T E'
E' → - T E' | ε
T  → F T'
T' → / F T' | ε
F  → id | num | ( E )
```

Input: `y = ( id - num ) / id`

Problem 6

```
S  → id = E
E  → T E'
E' → + T E' | - T E' | ε
T  → F T'
T' → * F T' | / F T' | ε
F  → id | num
```

Input: `res = id * num + id`

Problem 7

```
E  → T E'
E' → + T E' | ε
T  → F T'
T' → * F T' | / F T' | ε
F  → ( E ) | id | num
```

Input: `( id * id ) + num`

Problem 8

```
E  → T E'
E' → + T E' | - T E' | ε
T  → F T'
T' → * F T' | / F T' | ε
F  → id | num | ( E )
```

Input: `id * ( id + num )`

Problem 9

```
S  → ( L ) | a
L  → S L'
L' → , S L' | ε
```

Input: `( a , a )`

Problem 10

```
S    → ( L ) | atom
L    → S L'
L'   → , S L' | ε
atom → id | a
```

Input: `( id , a )`

Problem 11

```
S    → ( L ) | atom
L    → S L'
```

```
L'   → , S L' | ε
atom → num | id | a
```

Input: `( num , id , a )`

Problem 12

```
E  → T E'
E' → < T E' | ε
T  → id | num | ( E )
```

Input: `id < num`

Problem 13

```
E  → T E'
E' → < T E' | ε
T  → id | num | ( E )
```

Input: `a < b < c`
(treat a,b,c as identifiers / id)

Problem 14

```
E  → T E'
E' → . T E' | ε
T  → "id" | "num"
```

Input: `"id" . "num"`

Problem 15

```
E  → T E'
E' → . T E' | ε
T  → "id" | "num" | ( E )
```

Input: `"id" . ( "num" )`

Part B — CFG Construction Problems

Instructions: For each language below, write a context-free grammar (CFG). If the language is not context-free, provide a short justification.

1. $L = \{ a^n b^n \mid n \geq 0 \}$
2. $L = \{ a^n b^m \mid n, m \geq 0 \}$
3. $L = \{ a^n b^n c^n \mid n \geq 0 \}$ ⚠ *Not context-free*
4. Palindromes over {a, b}
5. $L = \{ a^n b^n c^m \mid n, m \geq 0 \}$
6. $L = \{ a^n b^m c^m \mid n, m \geq 0 \}$
7. $L = \{ w \in \{a, b\}^* \mid w$ has even number of a's $\}$
8. $L = \{ w \in \{a, b\}^* \mid w$ has even number of b's $\}$
9. $L = \{ w \in \{a, b\}^* \mid w$ has odd number of a's $\}$
10. $L = \{ w \in \{a, b\}^* \mid w$ has odd number of b's $\}$
11. $L = \{ w \in \{a, b\}^* \mid$ number of a's = number of b's $\}$
12. $L = \{ a^n b^n c^m d^m \mid n, m \geq 0 \}$
13. $L = \{ w \in \{a, b\}^* \mid w$ starts and ends with the same symbol $\}$
14. $L = \{ w \in \{a, b\}^* \mid w$ ends with "ab" $\}$
15. $L = \{ w \in \{a, b\}^* \mid$ length of w is even $\}$
