Assignment-2

1830150

a) Design an equivalent left recursion free and left factored grammar of the following grammar G_1. The set of non terminals is {E, P} and the set of terminals is {'a', '(', ',', ')'}.

Context-free Grammar G_1'

$P \rightarrow (E) \mid a$

$E \rightarrow E, P \mid P$

b) Construct FIRST and FOLLOW sets for the non-terminals of the resulting grammar.

c) Construct LL(1) parse table for the resulting grammar.

d) Validate the following syntax through predictive parsing using the constructed parse table. (a, (a, (a, a)))

18301150

a) Given, Non terminal set is { E, P }
terminal set is { 'a', '(', ',', ')' }
Context free Grammar G
$P \longrightarrow (E) \mid a$
$E \longrightarrow E, P \mid P$

So, left factor gramman,
$P \longrightarrow (E) \mid a$
$E \longrightarrow PE'$
$E' \longrightarrow ,PE' \mid \varepsilon$

left recursion free,
$E \longrightarrow E.P \mid P$
So, $A \longrightarrow A\alpha \mid \beta$, to
$A \longrightarrow \beta A'$
$A' \longrightarrow \alpha A' \mid \varepsilon$
here, $\alpha = ,P$
$\beta = P$
So, $E \longrightarrow PE'$
$E' \longrightarrow ,PE' \mid \varepsilon$

18301150

b) for first FIRST,

First (P) → { (, a}

First (E) → { (, a}

First (E') → { , , ε}

for FOLLOW,

Follow (P) → { $, , ,) }

Follow (E) → {) }

follow (E') ⇒ {) }

c) for P → (E)

First ((E)) → { (}

for P → a

First (a) → { a }

for, E → PE'

First (PE') → { (, a }

for, E' → PE'

for, E' → PE'

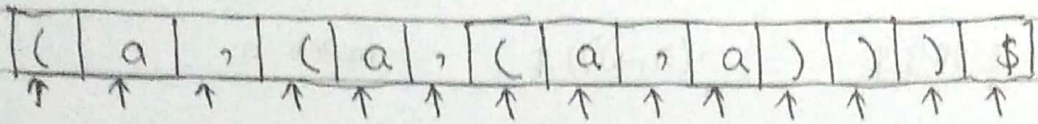
First (, PE') → { , }

	a	(	,	)	$
P	P→a	P→(E)			
E					
E'			E'→PE'	E'→ε	

LL(1) Parse table

18301150

d) Given, (a, (a, (a,a)))



Head	Stack	Input	Action
	P $	(a,(a,(a,a)))$	
	(E)$	(a,(a,(a,a)))$	P → (E)
(	E)$	a,(a,(a,a)))$	match (
(	PE')$	a(a,(a,a)))$	E → PE'
(	aE')$	a,(a,(a,a)))$	P → a
(a	E')$	,(a,(a,a)))$	match a
(a	,PE')$	,(a,(a,a)))$	E' → PE'
(a,	PE')$	(a,(a,a)))$	match
(a,	(E)E')$	(a,(a,a)))$	P → (E)
(a,(	E)E')$	a,(a,a)))$	match (
(a,(	PE')E')$	a,(a,a)))$	E → PE'
(a,(	aE')E')$	a,(a,a)))$	P → a

18301150

Head	Stack	Input	Action
(a,(a	E')F') $	,(a,a))) $	match a
(a,(a	,PE')E')$	,(a,a)))$	F' → , PE'
(a,(a,	PE')E')$	(a,a)))$	match ,
(a,(a,	(E)E')E')$	(a,a)))$	P → (E)
(a,(a,(	E)E')E')$	a,a)))$	match (
(a,(a,(	PE')E')E')$	a,a)))$	E → PE'
(a,(a,(	aF')F')E')$	a,a)))$	P → a
(a,(a,(a	E')E')E')$	,a)))$	match a
(a,(a,(a	,PE')E')E')$	,a)))$	E' → , PE'
(a,(a,(a,	PE')E')E')$	a)))$	match ,
(a,(a,(a,	aE')E')E')$	a)))$	P → a
(a,(a,(a,a	E')E')E')$	)))$	match a
(a,(a,(a,a	E)E')E')$	)))$	E' → ε
(a,(a,(a,a	)E')E')$	)))$	ε vanished

18301150

Head	Stack	Input	Action
(a,(a,(a,a)	E')E')$	))$	match)
(a,(a,(a,a)	ε)E')$	))$	E → ε
(a,(a,(a,a)	)E')$	))$	ε vanished
(a,(a,(a,a))	E')$	)$	match)
(a,(a,(a,a))	ε)$	)$	E' → ε
(a,(a,(a,a))	)$	)$	ε vanished
(a,(a,(a,a)))	$	$	match)

Therefore, the given syntax is valid