27%

	a	b	$
A	aBa		
B	@	bB	

Output

k =
row =
col =
flag =

Prod

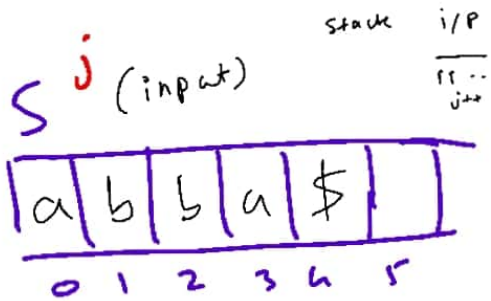
0	A → a B a
1	B → b B
2	B → @

table

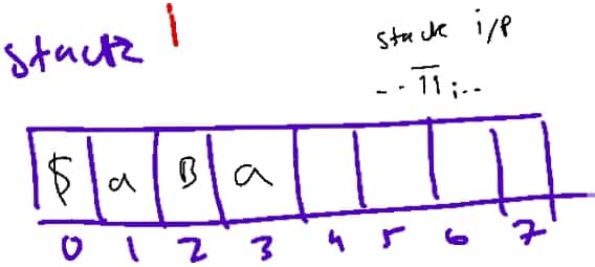
	0	1	2
0	a B a		
1	@	b B	

Size

	0	1	2
0	3	0	0
1	1	2	0



stack i



58 / 102

Pgm 3

Generate LL(1) parsing for the grammar

A → aBa
B → bB
B → @ (ε)

i/p assbag$

Steps to follow for LL(1)

1. Left Recursion ✗
2. Left factoring ✗
3. FIRST and follow ✓
4. Parsing table
5. stack

FIRST and FOLLOW

	FIRST	FOLLOW
A → aBa	a	$
B → bB | ε	b, ε	a

Parsing table

	a	b	$
A	aBa		
B	@	bB	

display(i, j)

=> print Stack(j) → stack(i)

=> print s(j) → s(n)

// stack contains current starting

// s contains i/p string

Variables

n = 5

i = 1

j = 0

k =

row =

col =

Code Flow

Stdio
Stalib
table prod
size
s
stack
Header Files (2)
Global variables (5 arrays)
Functions (2)

main()
=> Output the grammar (stored in prod)
=> Output the Predictive Table (stored in table)

	a	b	$
A	0	0	0
B	0	0	0

display(i, j)
=> print Stack(i) → stack(i)
=> print s(j) → s(n)
// stack contains current string
// s contains i/p string

Generate LL(1) parsing for the grammar

A → aBa
B → bB
B → @ (ε)

i/p aba

Steps to follow for LL(1)
1. Left Recursion ✗
2. Left factoring ✗
3. FIRST and Follow ✓
4. Parsing table
5. stack

FIRST and Follow

	FIRST	FOLLOW
A → aBa	a	$
B → bB | ε	b, ε	a

Parsing table

	a	b	$
A	aBa		
B		bB	

main()

=> Output the grammar (stored in prod)

=> Output the Predictive Table (stored in table)

	a	b	$
A	O	O	O
B	O	O	O

=> Get the i/p string

store in s

Get the stack initalized by $A

=> Output as

Stack Input

displayU=> each iter

stack()	s()
$A	abba$
$aBa	abba$

if match!

if same stack()=s()

Switch choose row

Switch choose col eg:- $aB bba

row col

if → 'U' ERROR (not defined Production)

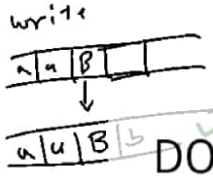
else-if → ε(@) → i-- just like ε substitution

else → substitute in reverse order

eg:- $aB bba$

B->bB

but in stack write



UP DOWN