CS & IT ENGINEERING

Compiler Design

Lexical Analysis & Syntax Analysis

DPP Discussion

Mallesham Devasane Sir

01 Question

02 Discussion

[NAT]

[1 Marks]

Consider the following C program:

```
int main ( )
{
/*finding maximum element out of a & b*/
int a, b, max;
a = 10; b = 20;
if (a < b)
    max = b;
else
    max = a;
return (max);
}
```

=41

Calculate the total number of tokens present in the program?

[MSQ]

[1 Marks]

Consider the following C-program:

```
1. int main )(
2. {
4. x = a + b* c;
5. y = x + a ;
6. char f = 'e' ;
7. in t g = 200;
8. ch/* comment ar = "gate";
9. }
```

Which of the following is correct regarding above program?

A. The given program has 47 tokens.

B. Given program produces compilation error

C. Given program produces lexical error

D. No error produced by program

[MCQ]

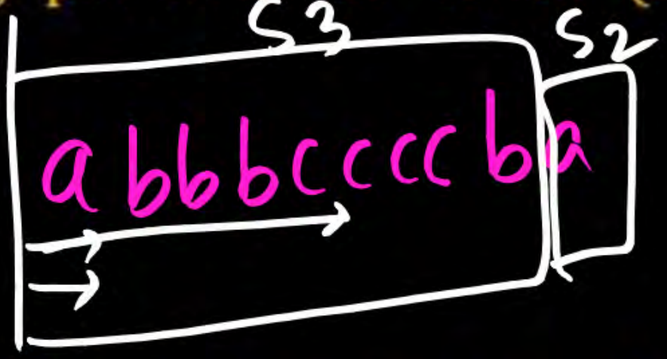
[1 Marks]

Compiler's first phase makes uses of following patterns for token (S_1,S_2,S_3) recognition over the alphabet a,b,c.

S_1: b#(b|a)*c = $(\varepsilon+b)(b+a)^*c$

S_2: c#(c|b)*a = $(\varepsilon+c)(c+b)^*a$

S_3: a#(b|c)*b = $(\varepsilon+a)(b+c)^*b$



Note: x# means 0 or 1 occurrence of the symbol x. The analyzer outputs the token that matched the longest possible prefix of the string. If abbbccccba is processed by first phase of compiler then which one of the following is the sequence of token of output.

A S_1, S_2, S_3

B S_1, S_2

C S_3, S_2

D S_3, S_3

[NAT]

[1 Marks]

How many of the following strings are said to be tokens in C-language without looking at next input character?

(i) ;

(ii) return

(iii) int

(iv) (

(v) &&

(vi) >>

= 2

[NAT]

[1 Marks]

Consider the following C-program:

```
int main ( )
{
  int x; /* comment */
  x = = = y /* abcd*** */ *abcd*/;
  int **p;
  int b = 10, y;
  x = *p ++ + ++ y;
}
```

How many tokens are present in the given program?

[MCQ] **[1 Marks]**

Which of the following is equivalent unambiguous grammar for following rules?

Operator	Priority	Associativity
↑, #	3	Left to right
⊕, *	2	Right to left
–, =	5 (Highest)	Left to right
/, &	1 (Lowest)	Right to left
+, $	4	Left to right

Note: 5 has the highest priority and 1 has the least priority.

(A)

$A \rightarrow B \uparrow A \mid B \# A \mid B$

$B \rightarrow C \oplus B \mid C * B \mid C$

$C \rightarrow C - D \mid C = D \mid D$

$D \rightarrow D + E \mid D \$ E \mid E$

$E \rightarrow E - F \mid E = F \mid F$

$F \rightarrow id$

(B)

$A \rightarrow A \uparrow B \mid A \# B \mid B$

$B \rightarrow C \oplus B \mid B * C \mid C$

$C \rightarrow C - D \mid C = D \mid D$

$D \rightarrow D + E \mid D \$ E \mid E$

$E \rightarrow F - E \mid F = E \mid F$

$F \rightarrow id$

(C)

$A \rightarrow A / B \mid A \& B \mid B$

$B \rightarrow B \oplus C \mid B * C \mid C$

$C \rightarrow C \uparrow D \mid C \# D \mid D$

$D \rightarrow D + E \mid D \$ E \mid E$

$E \rightarrow F - E \mid F = E \mid F$

$F \rightarrow id$

(D)

$A \rightarrow B / A \mid B \& A \mid B$

$B \rightarrow C \oplus B \mid C * B \mid C$

$C \rightarrow C \uparrow D \mid C \# D \mid D$

$D \rightarrow D + E \mid D \$ E \mid E$

$E \rightarrow E - F \mid E = F \mid F$

$F \rightarrow id$

[MCQ]

[2 Marks]

What will be the equivalent grammar after ~~reducing~~ applying left factoring ~~from~~ in the given grammar?

$S \rightarrow a|ab|abc|abcd|e|f$

$S \rightarrow e|f|aS'$

$S' \rightarrow \varepsilon|b|bc|bcd \Rightarrow S' \rightarrow \varepsilon|bA'$

$A' \rightarrow \varepsilon|c|cd \Rightarrow A' \rightarrow \varepsilon|cB'$

$B' \rightarrow \varepsilon|d$

(A) $S \rightarrow aS'$
$S \rightarrow b|c|d|e|f$

(B) $S \rightarrow e|f|S'$
$S' \rightarrow a|ab|abc|abcd$

(C) $S \rightarrow e|f|aS'|\in$
$S' \rightarrow bA'|\in$
$A' \rightarrow cB'|\in$
$B' \rightarrow d$

(D) $S \rightarrow e|f|aS'$
$S' \rightarrow bA'|\in$
$A' \rightarrow cB'|\in$
$B' \rightarrow d|\in$

[MSQ]

[2 Marks]

Which of the following is correct regarding FIRST & FOLLOW of the given grammar.

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \in$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \in$

$F \rightarrow id \mid (E)$

A. FIRST (T)= {id, (}
FOLLOW(T') = {+, $,)}

B. FIRST (E) = {id, (}
FOLLOW(E') = {$,)}

C. FIRST (E) = {id, (}
FOLLOW(F) = {+, $,), *}

D. FIRST (T') = {*, ∈}
FOLLOW (E) = {$, * ,)}

[NAT]

[2 Marks]

Consider the following grammar:

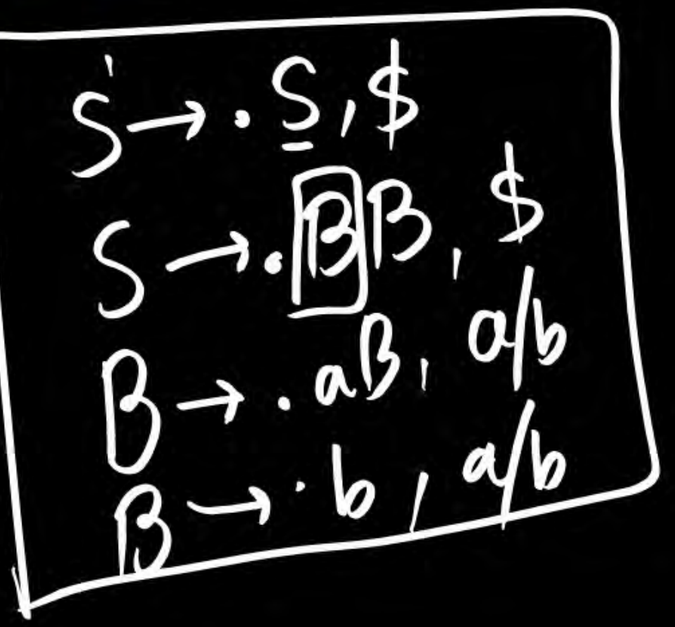
S→ BB

B→ aB|b

How many items are there in Closure (S' →.S, $)?

= 4

=



[NAT] **[2 Marks]**

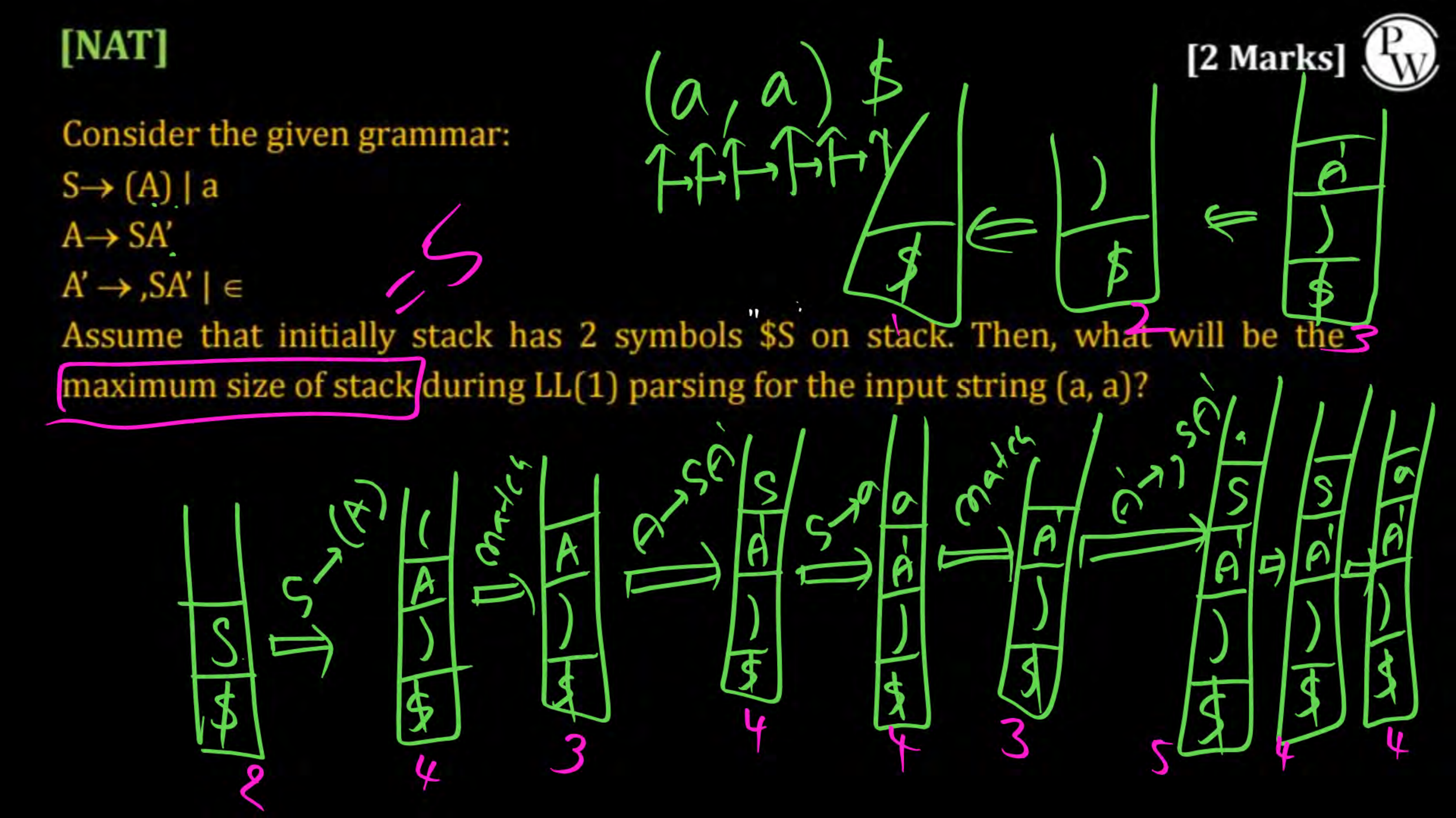
Consider the given grammar:

$S \rightarrow (A) \mid a$

$A \rightarrow SA'$

$A' \rightarrow ,SA' \mid \in$

Assume that initially stack has 2 symbols "$S on stack. Then, what will be the maximum size of stack during LL(1) parsing for the input string (a, a)?



[MCQ] [2 Marks]

Consider the following

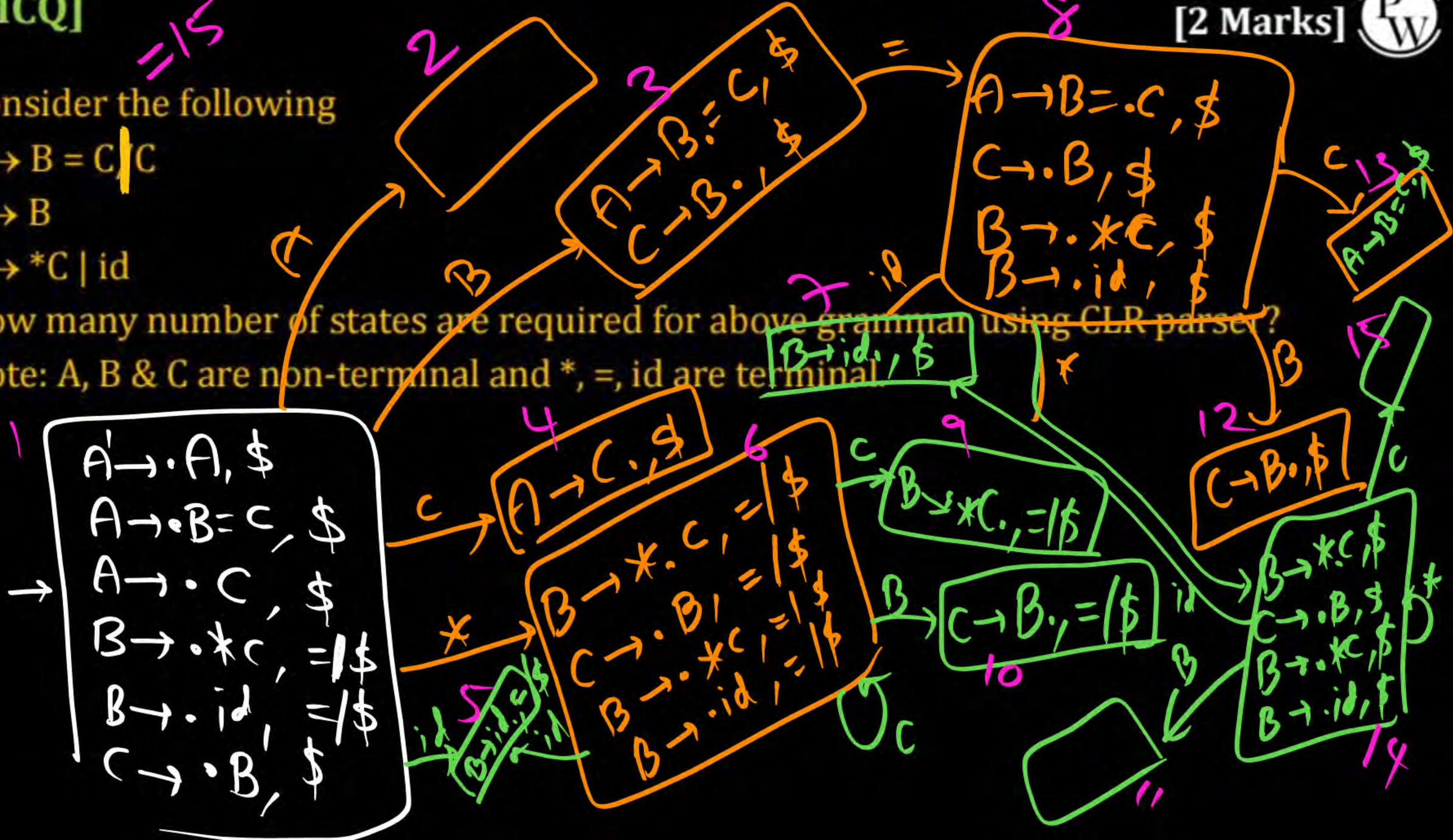
A → B = C | C

C → B

B → *C | id

How many number of states are required for above grammar using CLR parser?

Note: A, B & C are non-terminal and *, =, id are terminal.

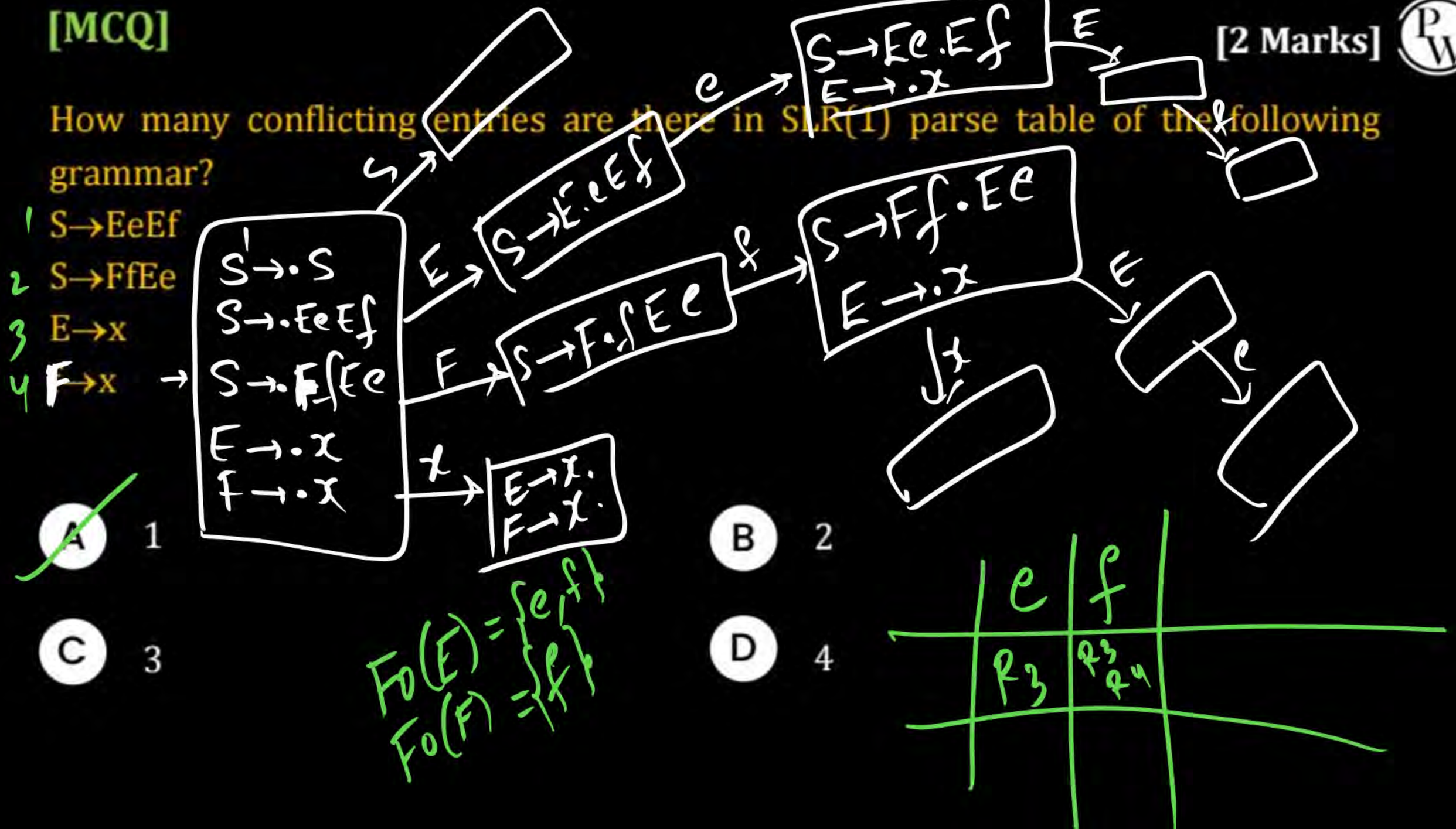


[MCQ]

[2 Marks]

How many conflicting entries are there in SLR(1) parse table of the following grammar?

1 S→EeEf
2 S→FfEe
3 E→x
4 F→x



A 1

B 2

C 3

D 4

Fo(E) = {e, f}
Fo(F) = {f}

	e	f
	R_3	R_3 R_4

[MSQ] **[2 Marks]**

Consider following grammar G.

S→ Aa| Bb

A→ Ac| ∈

B→ Bc| ∈

Which of the following statement is true?

$S' \to .S$

$S \to .Aa$

$S \to .Bb$

$A \to .Ac$

$A \to .$

$B \to .Bc$

$B \to .$

$Fo(A) = \{a, c\}$

$Fo(B) = \{b, c\}$

RR conf in LR(0)

$Fo(A) \cap Fo(B)$ is not empty

RR conf in SLR(1)

(A) It has shift-reduce conflict in the first state of the LR(0) machine.

(B) It has reduce-reduce conflict in the first state of the SLR(1) machine.

(C) It has reduce-reduce conflict on the first state on the LR(0) machine.

(D) It has shift-reduce conflict in the first state of the SLR(1) machine.

[MSQ]

[2 Marks]

Consider the following grammar:

P→ Qx | yQz | tz | ytx

Q→ t

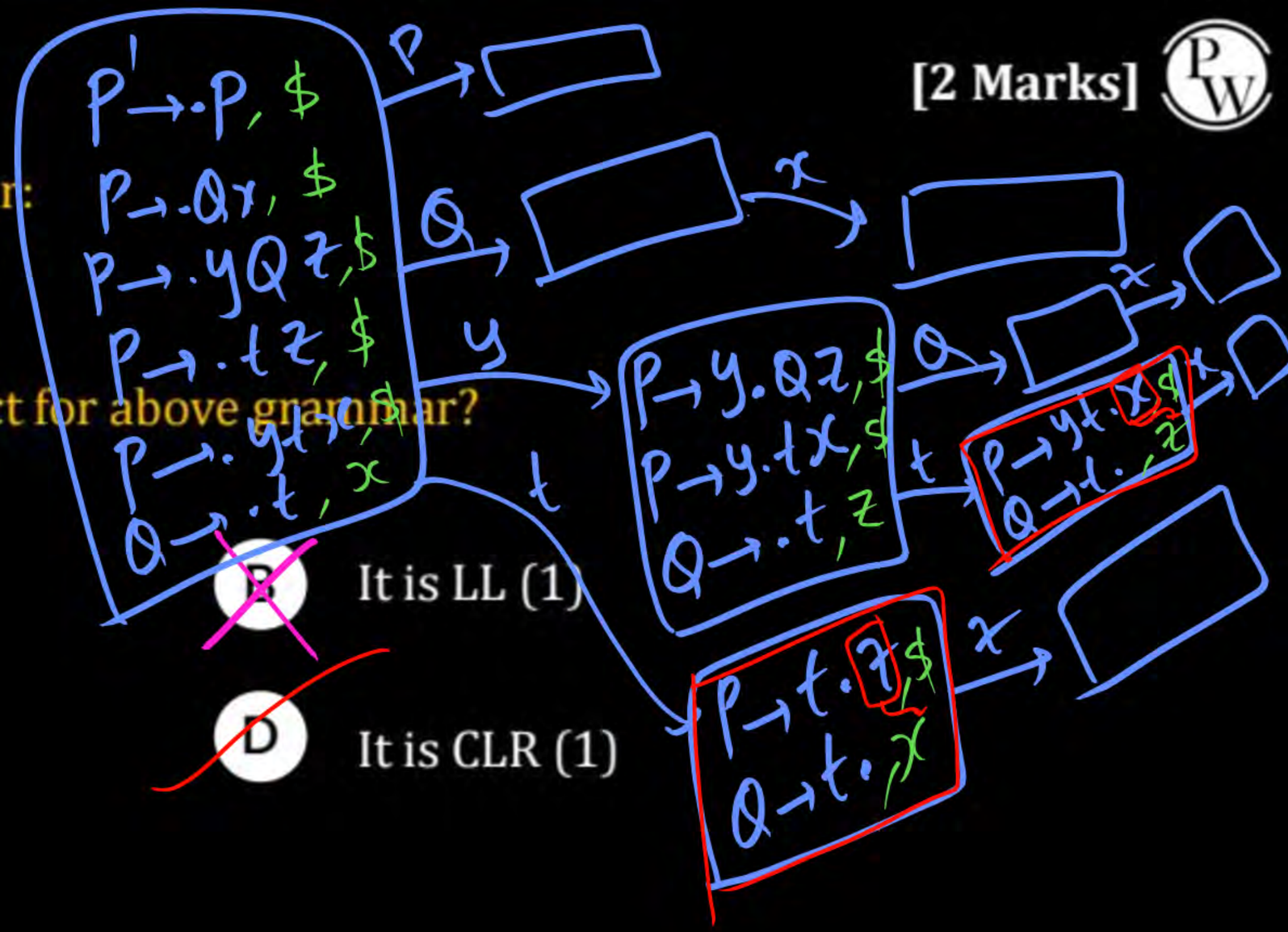
Which of the following is correct for above grammar?

A It is LR(1)

B It is LL (1)

C It is LALR(1)

D It is CLR (1)



[NAT] **[2 Marks]**

$\Rightarrow$ 20 Reduced moves

The maximum number of reduce moves that can be taken during bottom-up evaluation of 21 token string by using a bottom-up parser. Assuming the grammar has no epsilon and unit production.

$A \to \varepsilon$ $A \to B$ $A \to a$

No. of substitutions in RMD = No. of Reduced moves in Reverse of RMD

$w = xy$

1 Reduced move

$w = xyz$

$S \Rightarrow Az \Rightarrow xyz$

2 Reduced


THANK YOU GW
SOLDIERS !