American International University- Bangladesh (AIUB)
Faculty of Engineering (EEE)

Course Name:	**Compiler Design**		
Semester:	Fall (2021-22)	**Sec:**	G
Course Code	01035	**Faculty**	Masum Billah
Student Name:	**Safkat Jaman**	**Student ID:**	**19-40286-1**
Submission Date:	**10/12/21**		

(19-40286-1) Ans. to the Q-no: 1

Grammar,

S -> U

U -> TaU

U -> TaT

T -> aTbT

T -> bTaT

T -> d

A grammar is LL(1) when it's parsing table has no multiple entities.

Given Step	First	Follow
S -> U	{a, b, d}	{$}
U -> TaU | TaT	{a, b, d}	{a, b, $}
T -> aTbT | bTaT | d	{a, b, d}	{$}

Parsing table on basis of first and follow:

	a	b	d	$
S	U	U	U	The given grammar is not LL(1) because 2 entries are detected
T	T-> aTbT	T->bTaT	T -> d	
U	U-> TaT	U->TaT	U->TaT	
	U-> TaV	V->TaV	U->TaV	

Ans to the Q no: 2

(i) S -> 0A | 1B

A -> 0AA | 1S | 1

B -> 1BB | 0S | 0

Given step	First	follow
S->0A | 1B	{0,1}	{$}
A->0AA | 1S | 1	{0,1}	{$}
B-> 1BB | 0S | 0	{1,0}	{$}

	0	1	$
S	S->0A	S->1B	
A	A->0AA	A->1S A->1	
B	B->0S B->0	B->1BB	

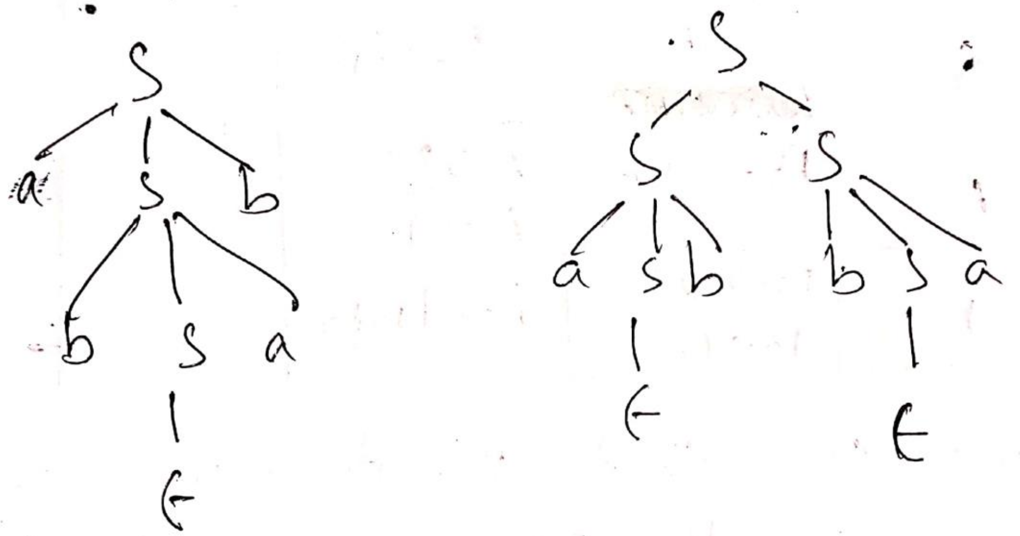
We can see that 2 entries are in the same cell. It means that it may generate one or more parse tree. So we can say that this grammar is ambigious.

~~② S -> a abS~~

② S -> aSb | bSa | SS | ε

	a	b
S->aSb| bSA| SS|ε	S->aSb S->bSA SS	S->aSb S->bSa SS

possible parsing tree



Here 2 parse trees possible so the grammar is ambiguous grammar.

(3) $S \rightarrow 1s1 \mid T$

$T \rightarrow 1X1 \mid X$

$X \rightarrow 0X0 \mid 1$

	first	Follow
$S \rightarrow 1s1 \mid T$	{1,0}	{\$}
$T \rightarrow 1X1$	{1,0}	{\$}
$X \rightarrow 0X0$	{1,0}	{\$}

	0	1	$
	S->T T->0X0 X->0x0	S->1s1 T->1x1 0x0 X->1	

2 entries found so the grammar is ambigouos.

Ans.to.the.Q.no:3

Construction of LL(1):

Step 1 : find first () and follow() function.

Step 2: Construct parse table

Step 3: Stack Implementation.

Step4: Parse the input string.

finding first() and follow():

-> first () and follow() set's are needed

So that the parser can properly apply the needed production.

(1) First ():

First (·) is a set of terminal symbol that begins in the strings derived for α,

$A \to aBc \mid dFg$

then first$(A) = \{a, d\}$

Rules for creating first() function:

(1) For a production rule $\to$ $X \to \epsilon$

First $(X) = \{\epsilon\}$

(2) For any terminal symbol

First {'terminal'} = {'symbol'}

(3) For a Production rule

$X \to Y_1 Y_2 Y_3$

calculating first(X):

⊛ If $\epsilon \notin$ first (Y_1) then first (X) = first(Y_1)

⊛ If $\epsilon \in$ first (Y_1) then first $(X) = \{$first$(Y_1) - \epsilon\}$ $\cup \{$first $(Y_2 Y_3)\}$

⊛ If $\epsilon \notin$ first (Y_2) then first $(Y_2 Y_3)$ = first (Y_2)

If $\epsilon \in$ first (Y_2) then first $(Y_2 Y_3) = \{$first(Y_2) $- \{\epsilon\}\} \cup \{$first $(Y_3)\}$

follow():

follow is a set of terminal that appear immediatly to the right of α

Rules of follow():

① for the start symbol s, place \$ in follow (s)

(2) For any production rule $A \to \alpha B$

$Follow(B) = Follow(A)$.

(3) For any production rule

$A \to \alpha B b$

If $\epsilon \notin First(B)$ then $Follow(B) = First(B)$

If $\epsilon \in First(B)$ then $Follow(B) = \{First(B) - \epsilon\} \cup \{Follow(A)\}$

For the following grammar the LL(1) parsing table and stack implementation of string 'acdb':

$S \to aABb$

$A \to c \mid \epsilon$

$B \to d \mid \epsilon$

First (S) = {a}

First (A) = {c}, {ϵ}

First (B) = {d}, {ϵ}

Follow (S) = {$}

Follow (A) = {d, b}

Follow (B) = {b}

Parsing table:

	a	b	c	d	$
S->aABb	S->aABb				
A		A->ϵ	A->c	A->ϵ	
B		B->ϵ		B->d	

Stack output of input

'acdb'

Stack	Input	Moves
$S	acdb$	S -> aABb
$bBA~~a~~	~~a~~cdb$	A -> c, pop
$bB~~c~~	~~c~~db$	B -> d, pop
$b~~d~~	~~d~~b$	pop'd'
$~~b~~	~~b~~$	pop(b)
$	$	accepted.