1- Give the First and Follow sets for the following grammar:

A → BCD
B→b|€
C→c | €
D→d

First(A)= { b,c,d}
First(B)={b,€}
First(C)={c,€}
First(D)= {d}

Follow(A)={$}
Follow(B)={c,d}
Follow(C)={d}
Follow(D)={$}

#2 and #3 Solutions (*):

2 –

a)

Solution:

1- Find First and Follow sets.
2- Fill the parsing table cells by placing the production in the first of that symbol. If € in the first, place the production in the Follow.

First(E)= {(,id}   //from E->TE'
Follow(E)={$,)}
First(E')={+, €}   //+ from E'->+TE', € from E'->€
Follow(E')={$,)}
First(T)={(,id}   //from T->FT'
Follow(T)={+,),$}
First(T')={*, €}   //* from T'->*FT' , € from T'->€
Follow(T')={+,),$}
First(F)={(,id}   // (from F->(E), id from F-> id
Follow(F)={*,+,),$}

First(S) = {(, a}

Follow(S) = {$, , ,)} //€ never comes in the follow set

NON-TERMINAL	INPUT SYMBOL					
	Id	+	*	(	)	$
E	E→TE’			E→TE’		
E’		E’→+TE’			E→€	E→€
T	T→FT’			T→FT’		
T’		T’→€	T’→*FT’		T’→€	T’→€
F	F→ id			F->(E)		

Parsing Table

3- b) S -> (L) | a And L -> L, S | S with input ((a,a),a,(a))

First(S)={(,a}
Follow(S)={$,’,’,)}
First(L)={(,a}
Follow(L)={,,)}

Production	Predict
S → (L)	{(}
S→ a	{a}
L→L,S	{(,a}
L→S	{(,a}

This grammar is not LL(1) because of Non-Disjoint Predict Sets.

Convert to LL(1) Grammar:

- No common Factor
- Eliminating Left Recursion

S → (L) | a

L → SL’

L’ → ,SL’|€

Check if it is LL(1):

S → (L) | a

L → SL’

L’ → ,SL’|€

First(L) = {(, a}

Follow(L) = {)}

First(L’)={€, , }

Follow(L’)={)}

If rhs =€, then add follow of the lhs to the predict set

Production	Predict
S → (L)	{(}
S→a	{a}
L→SL’	{(,a}
L‘→,SL’	{,}
L’→€	{)}
This grammar is LL(1) because of Disjoint Predict Sets.	

NT/T	(	)	,	a	$
S	S → (L)			S → a	
L	L→SL’			L→SL’	
L’		L’→€	L’ → ,SL’		

Parsing Table

Stack	Input	Action
$S	((a , a) , a , (a)) $	S → (L)
$)L(	((a , a) , a , (a)) $	Match (
$)L	(a , a) , a , (a)) $	L→SL’
$)L’S	(a , a) , a , (a)) $	S → (L)
$)L’)L(	(a , a) , a , (a)) $	Match (
$)L’)L	a , a) , a , (a)) $	L→SL’
$)L’)L’S	a , a) , a , (a)) $	S→a
$)L’)L’a	a , a) , a , (a)) $	Match a
$)L’)L’	, a) , a , (a)) $	L’→,SL’

$)L')L'S,	, a) , a , (a)) $	Match ,
$)L')L'S	a) , a , (a)) $	S→a
$)L')L'a	a) , a , (a)) $	Match a
$)L')L'	) , a , (a)) $	L'→€
$)L')	) , a , (a)) $	Match)
$)L'	, a , (a)) $	L'→,SL'
$)L'S,	, a , (a)) $	Match ,
$)L'S	a , (a)) $	S→a
$)L'a	a , (a)) $	Match a
$)L'	.(a)) $	L'→,SL'
$)L'S,	,(a)) $	Match ,
$)L'S	(a)) $	S→ (L)
$)L')L(	(a)) $	Match (
$)L')L	a)) $	L→SL'
$)L')L'S	a)) $	S→a
$)L')L'a	a)) $	Match a
$)L')L'	)) $	L'→€
$)L')	)) $	Match)
$)L'	) $	L'→€
$)	) $	Match)
$	$	Accept
Table Driven Predictive Parser		

*** Book:** "Compilers Principles, techniques, & tools", Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffry D. Ullman