Exercise 3

1. Given a following grammar:

   E ⟶ F T,

   T ⟶ AD F T,

   T ⟶ e,

   AD ⟶ +,

   AD ⟶ -,

   F ⟶ L K,

   K ⟶ MUL L K,

   K ⟶ e,

   MUL ⟶ *,

   MUL ⟶ /,

   L ⟶ (E),

   L ⟶ id

   a. Find the first and follow sets of the grammar.

   b. The parsing table of the grammar

2. From the parsing table in (1), use stack to simulate leftmost derivation as the LL(1) parsing for stream of tokens id + id * (id + id).

No.	Stack	Tokens	Action
1.	$	id + id * (id + id)$	
2.			
…	…	…	…

3. Given a following grammar:

E ⟶ E AD F,

E ⟶ F,

AD ⟶ +,

AD ⟶ -,

F ⟶ F MUL L ,

F ⟶ L,

MUL ⟶ *,

MUL ⟶ /,

L ⟶ (E),

L ⟶ id

   a. Is the grammar LL(1)? Justify your answer.
   b. If it’s not LL(1), how to change the grammar to LL(1)?

4. The following is a grammar for regular expressions over symbols a and b only, using + in place of | for union, to avoid conflict with the use of vertical bar as a metasymbol in grammars:

```
rexpr -> rexpr + rterm | rterm
rterm -> rterm rfactor | rfactor
rfactor -> rfactor * | rprimary
rprimary -> a | b
```

   a. Left factor this grammar.
   b. Does left factoring make the grammar suitable for top-down parsing?
   c. In addition to left factoring, eliminate left recursion from the original grammar.
   d. Is the resulting grammar suitable for top-down parsing? Justify your answer.