4.7 Given the grammar $A \rightarrow (A) A \mid \varepsilon$,

a. Construct First and Follow sets for the nonterminal A.

b. Show this grammar is LL(1).

4.8 Consider the grammar

lexp → *atom* | *list*
atom → **number** | **identifier**
list → **(** *lexp-seq* **)**
lexp-seq → *lexp-seq lexp* | *lexp*

a. Remove the left recursion.

b. Construct First and Follow sets for the nonterminals of the resulting grammar.

c. Show that the resulting grammar is LL(1).

d. Construct the LL(1) parsing table for the resulting grammar.

e. Show the actions of the corresponding LL(1) parser, given the input string **(a (b (2)) (c))**.

4.9 Consider the following grammar (similar, but not identical to the grammar of Exercise 4.8):

lexp → *atom* | *list*
atom → **number** | **identifier**
list → **(** *lexp-seq* **)**
lexp-seq → *lexp* **,** *lexp-seq* | *lexp*

a. Left factor this grammar.

b. Construct First and Follow sets for the nonterminals of the resulting grammar.

declaration → *type var-list*
type → **int** | **float**
var-list → **identifier,** *var-list* | **identifier**

a. Left factor this grammar.

b. Construct First and Follow sets for the nonterminals of the resulting grammar.

Ejercicio 4.7 Louden

Primeros		
Regla	**Paso 1**	**Paso 2**
A → (A) A	(	
A → *e*	(, *e*	

Primeros(A)={(, *e*}

Siguientes		
Regla	**Paso 1**	**Paso 2**
A → (A) A	$,)	
A → *e*		

Siguientes(A)={$,)}

Ejercicio 4.8 Louden

Gramática original

lexp → atom | list
atom → number | identifier
list → (lexp-seq)
lexp-seq → lexp-seq lexp | lexp

Eliminación de la recursividad por la izquierda:

lexp → atom | list
atom → number | identifier
list → (lexp-seq)
lexp-seq → lexp lexp-seq | lexp

Originalmente eliminamos asi (como abajo) pero nos pareció que un estado estaba de más.

lexp → atom | list
atom → number | identifier
list → (lexp-seq)
lexp-seq → lexp lexp-seq'
lexp-seq' → lexp-seq | *e*

Primeros		
Regla	**Paso 1**	**Paso 2**
lexp → atom		number, identifier
lexp → list		number, identifier, (
atom → **number**	number	
atom → **identifier**	number, identifier	
list → **(**lexp-seq**)**	(	
lexp-seq → lexp lexp-seq		number, identifier, (
lexp-seq → lexp		number, identifier, (

Primeros(lexp)={number, identifier, (}

Primeros(atom)={number, identifier}

Primeros(list)={(}

Primeros(lexp-seq)={number, identifier, (}

Siguientes		
Regla	**Paso 1**	**Paso 2**
lexp → atom	**lexp:** $ **atom:** $	**atom:** $, number, identifier, (,)
lexp → list	**list:** $	**list:** $, number, identifier, (,)

Siguientes(lexp)={$, number, identifier, (,)}

Siguientes(atom)={$, number, identifier, (,)}

atom → **number**		
atom → **identifier**		
list → **(**lexp-seq**)**	**lexp-seq:**)	
lexp-seq → lexp lexp-seq	**lexp:** $, number, identifier, (,)	
lexp-seq → lexp	**lexp:** $, number, identifier, (,)	

Siguientes(list)={$, number, identifier, (,)}
Siguientes(lexp-seq)={)}

Ejercicio 4.9 Louden

Gramática original

lexp → atom | list
atom → number | identifier
list → (lexp-seq)
lexp-seq → lexp , lexp-seq | lexp

Factorización a la izquierda:

lexp → atom | list
atom → number | identifier
list → (lexp-seq)
lexp-seq → lexp lexp-seq'
lexp-seq' → , lexp-seq | *e*

Primeros		
Regla	**Paso 1**	**Paso 2**
lexp → atom		number, identifier
lexp → list		number, identifier, (
atom → **number**	number	
atom → **identifier**	number, identifier	
list → **(**lexp-seq**)**	(	
lexp-seq → lexp lexp-seq'		number, identifier, (
lexp-seq' → **,** lexp-seq	,	
lexp-seq' → ***e***	*e*	

Primeros(lexp)={number, identifier, (}

Primeros(atom)={number, identifier}

Primeros(list)={(}

Primeros(lexp-seq)={number, identifier, (}

Primeros(lexp-seq')={,, *e*}

Siguientes		
Regla	**Paso 1**	**Paso 2**
lexp → atom	**lexp:** $ **atom:** $	**lexp:** $, ,, *e* **atom:** $, ,, *e*
lexp → list	**lexp:** $ **list:** $	**lexp:** $, ,, *e* **list:** $, ,, *e*

Siguientes(lexp)={$, ,, *e*}

Siguientes(atom)={$, ,, e}

atom → **number**		
atom → **identifier**		
list → **(**lexp-seq**)**	**lexp-seq:**)	
lexp-seq → lexp lexp-seq'	**lexp:** $, ,, *e* **lexp-seq':**)	
lexp-seq' → **,** lexp-seq		
lexp-seq' → ***e***		

Siguientes(list)={$, ,, e}
Siguientes(lexp-seq)={)}
Siguientes(lexp-seq')={)}

Ejercicio 4.10 Louden

Gramática original

declaration → type var-list
type → int | float
var-list → identifier, var-list | identifier

Factorización a la izquierda:

declaration → type var-list
type → int | float
var-list → identifier var-list'
var-list' → , var-list | *e*

Primeros		
Regla	**Paso 1**	**Paso 2**
declaration → type var-list		int, float
type → **int**	int	
type → **float**	int, float	
var-list → **identifier** var-list'	identifier	
var-list' → **,** var-list	,	
var-list' → ***e***	*e*	

Primeros(declaration)={int, float}
Primeros(type)={int, float}
Primeros(var-list)={identifier}
Primeros(var-list')={,, *e*}

Siguientes		
Regla	**Paso 1**	**Paso 2**
declaration → type var-list	**declaration:** $ **type:** identifier **var-list:** $	
type → **int**		
type → **float**		
var-list → **identifier** var-list'	**var-list':** $	
var-list' → **,** var-list		
var-list' → ***e***		

Siguientes(declaration)={$}

Siguientes(type)={identifier}
Siguientes(var-list)={$}
Siguientes(var-list')={$}