Problem 1

Grammar:

`S ::= cSdS | dScS | ϵ`

1. Two leftmost derivations for the sentence `cdcd`

First:

$cSdS \rightarrow cdScSdS \rightarrow cd\epsilon cSdS \rightarrow cdc\epsilon dS \rightarrow cdcd\epsilon \rightarrow cdcd$

Second: $cSdS \rightarrow c\epsilon dS \rightarrow cdcSdS \rightarrow cdc\epsilon dS \rightarrow cdcd\epsilon \rightarrow cdcd$

2. Two rightmost derivations for the sentence `cdcd`

First:

$cSdS \rightarrow cSdcSdS \rightarrow c\epsilon dcSdS \rightarrow cdc\epsilon dS \rightarrow cdcd\epsilon \rightarrow cdcd$

Second:

$cSdS \rightarrow cdScSdS \rightarrow cd\epsilon cSdS \rightarrow cdc\epsilon dS \rightarrow cdcd\epsilon \rightarrow cdcd$

Problem 2

Grammar:

```
S ::= ( L ) | x
L ::= L , S | S
```

1. Left-most derivation of $(x,(x,x))$.

$S \to (L) \to (L, S) \to (S, S) \to (x, S) \to (x, (L)) \to (x, (L, S)) \to ($

2. Right-most derivation of $(x, (x, x))$.

$S \to (L) \to (L, S) \to (L, (L)) \to (L, (L, S)) \to (L, (L, x)) \to (L, ($

3. Show the steps that a shift-reduce parser goes through when it parses (x, x, x).

Stack	Input	Action
$\$$	$(x, x, x)\$$	shift
$\$($	$x, x, x)\$$	shift
$\$(x$	$, x, x)\$$	reduce by $S \to x$
$\$(S$	$, x, x)\$$	reduce by $L \to S$
$\$(L$	$, x, x)\$$	shift
$\$(L,$	$x, x)\$$	shift
$\$(L, x$	$, x)\$$	reduce by $S \to x$
$\$(L, S$	$, x)\$$	reduce by $L \to L, S$
$\$(L$	$, x)\$$	shift
$\$(L,$	$x)\$$	shift
$\$(L, x$	$)\$$	reduce by $S \to x$
$\$(L, S$	$)\$$	reduce by $L \to L, S$
$\$(L$	$)\$$	shift
$\$(L)$	$\$$	Reduce by $S \to (L)$
$\$S$	$\$$	reduce by $L \to S$
$\$L$	$\$$	accept

4. Swapping `L ::= L , S` production with `L ::= S , L` production.

Stack	Input	Action
$\$$	$(x,x,x)\$$	shift
$\$($	$x,x,x)\$$	shift
$\$(x$	$,x,x)\$$	reduce by $S \to x$
$\$(S$	$,x,x)\$$	shift
$\$(S,$	$x,x)\$$	shift
$\$(S,x$	$,x)\$$	reduce by $S \to x$
$\$(S,S$	$,x)\$$	shift
$\$(S,S,$	$x)\$$	shift
$\$(S,S,x$	$)\$$	reduce by $S \to x$
$\$(S,S,S$	$)\$$	reduce by $L \to S$
$\$(S,S,L$	$)\$$	reduce by $L \to S,L$
$\$(S,L$	$)\$$	reduce by $L \to S,L$
$\$(L$	$)\$$	shift
$\$(L)$	$\$$	reduce by $S \to (L)$
$\$S$	$\$$	reduce by $L \to S$
$\$L$	$\$$	accept

As we can see, the number of steps that the parser goes through remains the same. However, it will require more backtracking if an always reduce heuristic is used because making the reduction $L \to S$ on the first two S will result in a non-acceptance state.

Problem 3

Grammar:

```
Start ::= S

S ::= A a

A ::= B C | B C f

B ::= b

C ::= c
```

Parse table:

State	a	b	c	f	$\$$	A	B	C	S
0					acc				
1		s2	s3			g7	g4		
2	r4	r4	r4	r4					
3	r4	r4	r4	r4					
4			s5					g6	
5	r6	r6	r6	r6					
6	r1	r1	r1	s7					
7	r8	r8	r8	r8					
8									
9	r1	r1	r1	r1					

From this parse table we see that the grammar is an LR(1) grammar because there are no shift reduce conflicts. The only potential question would be when the stack has the values `BC`, in which case the question of whether to shift or reduce arises. But since we have a lookahead value of 1 we can see if the next token is `f`, in which case we shift, or anything else, in which case we reduce.

Problem 4

$\nabla\Delta$

Grammar:

$$\Big| := \nabla \Big|$$

```
S ::= ∇ S Δ | ϵ
```

Parse table:

State	∇	Δ	$\$$	S
0			acc	g1
1	s2			
2		r0		g1

From this parse table we see that there are no shift-reduce conflicts meaning that the grammar is LR(1).