Here are the answers for the practice problems.
   I strongly suggest that you DO the problems before you look at
   the solutions. If you just follow along the solutions without
   actually doing the problem yourself, you will have problems with
   the midterm.

P.S. If you notice any mistakes in the solutions, please send me email.

-------------------------------------------------------------------------
1   a) strings beginning and ending in 0
    b) all strings
    c) strings with 0 as third digit from right
-------------------------------------------------------------------------
2   a) 1*(0|01)*       // after any 0, only allow zero or one 1
    b) 1*0*(1|epsilon)0* // only a single 1 after first 0
-------------------------------------------------------------------------
3   a) NFA for RE = (a|b)*

(State 7 is the final state)

NFA for RE = (a*|b*)*

(State 11 is the final state)
b) Sequence of moves for DFA for (a|b)* in parsing "ababbab"

1,2,4,6, // a  
1,3,5,6, // b  
1,2,4,6, // a  
1,3,5,6, // b  
1,3,5,6, // b  
1,2,4,6, // a  
1,3,5,6,7 // b

Sequence of moves for DFA for (a*|b*)* in parsing "ababbab"

1,2,4,6,8,10, // a  
1,3,5,7,9,10, // b  
1,2,4,6,8,10, // a  
1,3,5,7,9,10, // b  
1,3,5,7,9,10, // b  
1,2,4,6,8,10, // a  
1,3,5,7,9,10,11 // b

Another sequence of moves for DFA for (a*|b*)* in parsing "ababbab"

1,2,4,6,8,10, // a  
1,3,5,7,9,10, // b  
1,2,4,6,8,10, // a  
1,3,5,7,9,10, // b  
1,3,5,7,9,10, // b  
1,2,4,6,8,10, // a  
1,3,5,7,9,10,11 // b

c) DFA for (a|b)* from NFA using subset construction

State 0                    =  { 1,2,3,7 }
State 1 = Move(State 0, a) =  { 1,2,3,4,6,7 }
State 2 = Move(State 0, b) =  { 1,2,3,5,6,7 }
Move(State 1, a) = State 1
Move(State 1, b) = State 2
Move(State 2, a) = State 1
Move(State 2, b) = State 2

S      ----                ----
t     //--\      a       //--\ <----
| 0|------->|| 1||      |a|
r     \--//              \--// ----
|a
|b------->|| 2||      |b|
\--// -----             ----
(State 0 is the start state)
(States 0,1,2 are all final states, since they include State 7 of the DFA)
DFA for (a*|b*)* from NFA using subset construction

State 0 = { 1,2,3,4,5,8,9,10,11 }
State 1 = Move(State 0, a) = { 1,2,3,4,5,6,8,9,10,11 }
State 2 = Move(State 0, b) = { 1,2,3,4,5,7,8,9,10,11 }
Move(State 1, a) = State 1
Move(State 1, b) = State 2
Move(State 2, a) = State 1
Move(State 2, b) = State 2

States 0, 1, 2 are all final states, since they include State 11 of the DFA

Initial Partition:
Partition 0 = { All final states } = { State 0, 1, 2 }
Partition 1 = { All nonfinal states } = { }

Refine Partition 0:

a
State 0 ---> State 1 (partition 1 to partition 1)
b
State 0 ---> State 2 (partition 1 to partition 1)
a
State 1 ---> State 1 (partition 1 to partition 1)
b
State 1 ---> State 2 (partition 1 to partition 1)
a
State 2 ---> State 1 (partition 1 to partition 1)
b
State 2 ---> State 2 (partition 1 to partition 1)

( Partition 0 can not be split further )
Resulting minimal DFA:

```
  --
 a| |
 S | V
 t ----
 a //--\ <----
r --> || 0|| |b
 t \--// -----
```

(State 0 is the start state. State 0 is also a final state)

4  a) Grammar for all strings of 0's and 1's with same number of 0's and 1's

\[
S \rightarrow A \ S \ B \mid B \ S \ A \mid \text{epsilon} \\
A \rightarrow 1 \ S \mid S \ 1 \\
B \rightarrow 0 \ S \mid 0 \ S \\
\]

or

\[
S \rightarrow 1 \ S \ 0 \ S \mid 0 \ S \ 1 \ S \mid \text{epsilon} \\
\]

Both grammars are ambiguous

b) Grammar for all strings of 0's and 1's with more 0's than 1's

\[
S' \rightarrow B \ S' \mid B \\
S \rightarrow A \ S \ B \mid B \ S \ A \mid \text{epsilon} \\
A \rightarrow 1 \ S \mid S \ 1 \\
B \rightarrow 0 \ S \mid 0 \ S \\
\]

Grammar is ambiguous

c) Grammar for all strings with balanced parentheses

\[
S \rightarrow S \ S \mid ( \ S ) \mid \text{epsilon} \\
\]

5  a) Parse trees for

(a,a)           (a,(a,a))

```
S   S
/\    /\  
( L ) ( L )
/\    /\  
L , S L , S
|    |  /\  
S   a S ( L )
|    |  /\  
a   a L , S
|    |  
S a
|  
a
```
b) \[ S \rightarrow (L) \rightarrow (L,S) \rightarrow (S,S) \rightarrow (a,S) \rightarrow (a,a) \]
\[ S \rightarrow (L) \rightarrow (L,S) \rightarrow (S,S) \rightarrow (a,S) \rightarrow (a,(L,S)) \rightarrow (a,(S,S)) \rightarrow (a,(a,S)) \rightarrow (a,(a,a)) \]

c) \[ S \rightarrow (L) \rightarrow (L,S) \rightarrow (L,a) \rightarrow (S,a) \rightarrow (a,a) \]
\[ S \rightarrow (L) \rightarrow (L,S) \rightarrow (L,(L)) \rightarrow (L,(L,S)) \rightarrow (L,(L,a)) \rightarrow (L,(S,a)) \rightarrow (L,(a,a)) \rightarrow (S,(a,a)) \rightarrow (a,(a,a)) \]

6  a) \[ S \rightarrow aSbS \rightarrow abS \rightarrow abaSbS \rightarrow ababS \rightarrow abab \]
\[ S \rightarrow aSbS \rightarrow abSaSbS \rightarrow abaSbS \rightarrow ababS \rightarrow abab \]

b) \[ S \rightarrow aSbS \rightarrow aSbaSbS \rightarrow aSbaSb \rightarrow aSbab \rightarrow abab \]
\[ S \rightarrow aSbS \rightarrow aSb \rightarrow abSaSb \rightarrow abSab \rightarrow abab \]

Example parse trees

```
S   S
 / | \   / | \ 
/ | | \   / | | \
 a S b \ a S b S
  |   S   / | \ 
  e / | \   b S a S
 / | | \   e 
 a S b S   e 
  |   e 
  e   e
```