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.

---

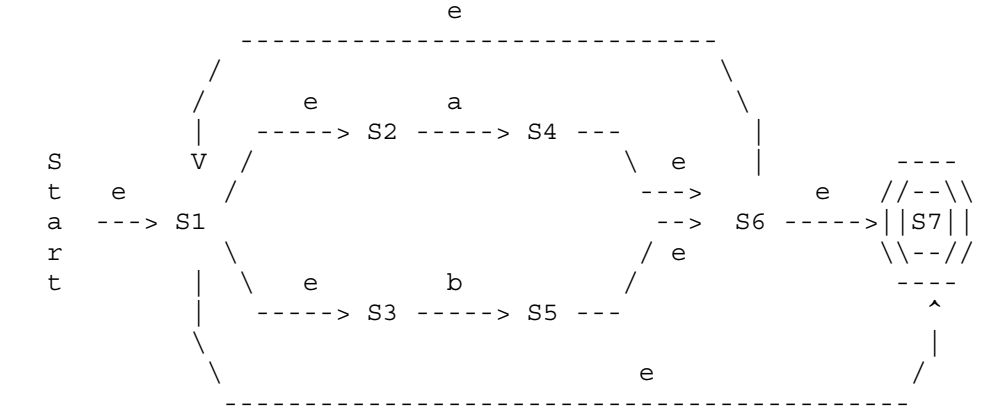
1 a) strings beginning and ending in 0
b) all strings
c) strings with 0 as third digit from right

---

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

---

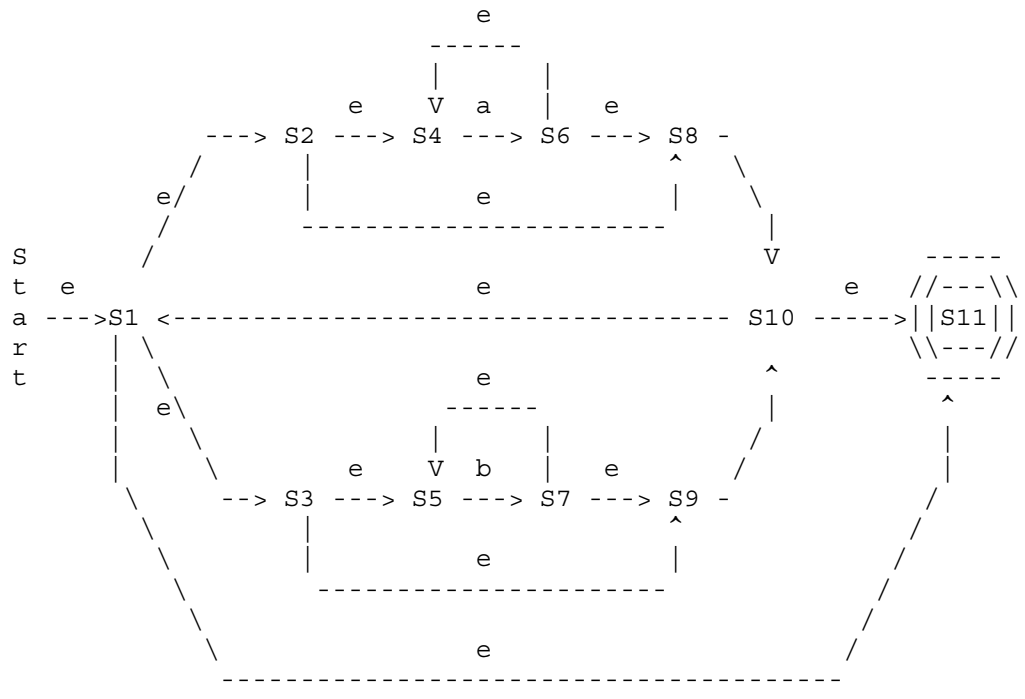
3 a) NFA for RE = (a|b)*



(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"

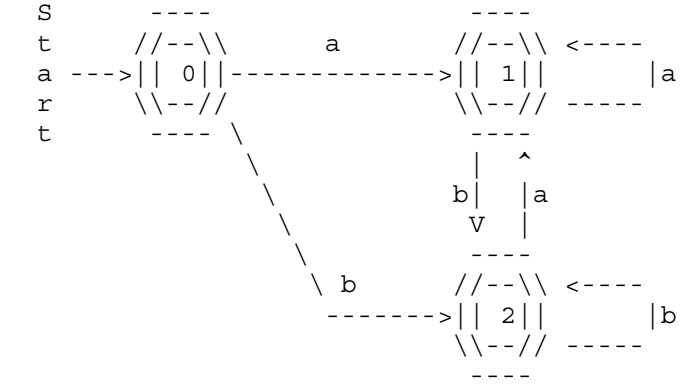
```
1,2,4,6,8,10,     // a
1,3,5,7,9,10,     // b
1,2,4,6,8,10,     // a
1,3,5,7,9,10,     // b
1,3,5,7,9,10,     // b
1,2,4,6,8,10,     // a
1,3,5,7,9,10,11   // b
```

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
1,3,5,7,5,79,10,  // b,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

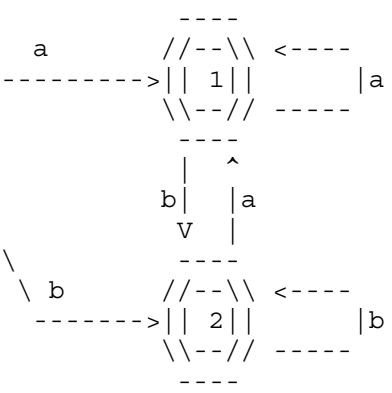
```
   State 0                     =  { 1,2,3,7 }
   State 1 = Move(State 0, a) =  { 1,2,3,4,6,7 }
   State 2 = Move(State 0, b) =  { 1,2,3,5,6,7 }
   Move(State 1, a) = State 1
   Move(State 1, b) = State 2
   Move(State 2, a) = State 1
   Move(State 2, b) = State 2
```



(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



(State 0 is the start state)
(States 0,1,2 are all final states, since they include State 11 of the DFA)

~~~~~~
d) Minimization of DFA for (a|b)* and (a*|b*)* [same 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)

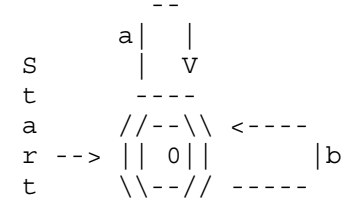
b
State 1 ---> State 2 (partition 1 to partition 1)

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:



(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 -> A S B | B S A | epsilon     // strings with same # of 0's & 1's
A -> 1 S | S 1                   // strings with one more 1 than 0
B -> 0 S | 0 S                   // strings with one more 0 than 1

     or

S -> 1 S 0 S | 0 S 1 S | epsilon // strings with same # of 0's & 1's
```

Both grammars are ambiguous

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

```
S' -> B S' | B                   // strings with more 0's than 1's
S -> A S B | B S A | epsilon     // strings with same # of 0's & 1's
A -> 1 S | S 1                   // strings with one more 1 than 0
B -> 0 S | 0 S                   // strings with one more 0 than 1
```

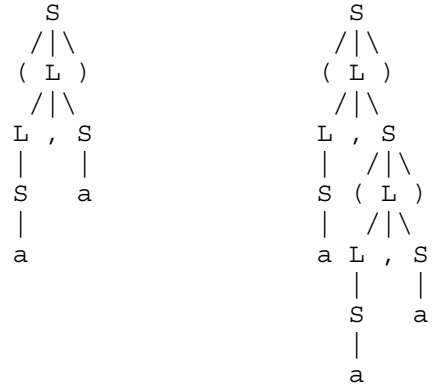
Grammar is ambiguous

c) Grammar for all strings with balanced parentheses

```
S -> S S | ( S ) | epsilon
```

---

5 a) Parse trees for



b) S -> (L) -> (L,S) -> (S,S) -> (a,S) -> (a,a)

S -> (L) -> (L,S) -> (S,S) -> (a,S) -> (a,(L)) -> (a,(L,S))
-> (a,(S,S)) -> (a,(a,S)) -> (a,(a,a))

c) S -> (L) -> (L,S) -> (L,a) -> (S,a) -> (a,a)

S -> (L) -> (L,S) -> (L,(L)) -> (L,(L,S)) -> (L,(L,a))
-> (L,(S,a)) -> (L,(a,a)) -> (S,(a,a)) -> (a,(a,a))

---

6 a) S -> aSbS -> abS -> abaSbS -> ababS -> abab
S -> aSbS -> abSaSbS -> abaSbS -> ababS -> abab

b) S -> aSbS -> aSbaSbS -> aSbaSb -> aSbab -> abab
S -> aSbS -> aSb -> abSaSb -> abSab -> abab

Example parse trees