1. Construct a DFA that recognizes each of the following languages. Justify your answer by describing, in English, the purpose of each state.

   a. The strings over \( \{x, y, z, \neg, \land, \lor, \bot, (, )\} \) that represent CNF formula in which every clause has one set of parenthesis around it and no other parenthesis are used.

   State 0 is the start state and also the accepting state for an empty formula.
   States 1-3 recognize literals connected by the or symbol.
   State 4 recognizes the end of a clause and is an accepting state.
   State 5 waits for the start of a new clause.
b. The strings over \{a,b\} in which the parities of a and b are the same.

State 0: we’ve seen an even number of a’s and b’s
State 1: we’ve seen an odd number of a’s and an even number of b’s
State 2: we’ve seen an odd number of a’s and b’s
State 3: we’ve seen an even number of a’s and an odd number of b’s

Both 0 and 2 are accepting states.

c. The strings over \{a,b\} in which each consecutive block of 5 symbols has at least two b’s.

The states of this machine are indexed by integers k,j where k is the number of symbols read since the last b and j is the number of symbols read since the second to last b.

From state (k,j) we transition to (k+1,j+1) on an a and (0,k+1) on b. All states shown are accepting states. There is an implicit dead state.

Finally, notice that a string x is in the language iff bbx is in the language. So we can start in state (0,1).
2. Construct an NFA that recognizes the following languages. Justify your answer by describing, in English, the purpose of each state.
   a. The strings over \{a,b\} containing at least 2 a’s and such that some pair of a’s is separated by a string whose length is a multiple of 3.
The transitions from states 0 to 1 and 4 to 5 verify two a’s. The transitions from state 1-4 verify that the a’s are separated by a string whose length is a multiple of 3.

b. The strings over \{a,b,c\} that have the same value when multiplied from the left as from the right according to the following (non-associative) multiplication table:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>

We first construct an NFA $M_a$ that accepts an input string provided it equals $a$ when multiplied from the left and from the right. The states of $M_a$ are $(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)$. In addition there is a start state $(\epsilon,a)$ and one final state $(a, \epsilon)$.

From state $(s_1,s_2)$ on input $t$ we transition to every state $(s_3, s_4)$ such that the $s_1 t = s_3$ and $t s_4 = s_2$ under the multiplication table. Additionally we define $\epsilon t = t \epsilon = t$.

For example there are transitions from $(\epsilon,a)$ on the symbol $a$ to states $(a,a)$, $(a,b)$, and $(a,\epsilon)$, on the symbol $b$ to the state $(b,b)$, and on the symbol $c$ to the state $(c,c)$.

Notice that under this construction, we maintain the following invariant: On input $x$ the machine can be in state $(s_1,s_2)$ iff $x$ equals $s_1$ when multiplied from the left and $x s_2$ equals the symbol $a$ when multiplied from the right. Therefore the machine can conclude in $(a, \epsilon)$ iff the input $x$ equals $a$ when multiplied from the left and the right.

We can construct similar machines $M_b$ and $M_c$. Our final machine $M$ has an accepting start state and $\epsilon$-transitions to the start states of $M_a, M_b$ and $M_c$. If the
input x is in the language then its product from the left and the right is some s \in \{a,b,c\} and x will have an accepting path in the sub-machine M_s. On the other hand, if there is an accepting path in one of the machines, x must have the same product when multiplied from the left and from the right, so the x in L.

3. Construct a regular expression for each of the following languages. Provide justification of correctness.
   a. The empty string and the set of strings over \{0,1\} that represent, in binary, a number that is equivalent to zero modulo 3. 
      \(((0+11)*(10(0+10)*11)^*)^*\)
      To see this transform the DFA (which we built in class) into a regular expression.
   b. The strings over \{a,b\} with an equal number of a’s and b’s such that in every prefix the number of a’s and the numbers of b’s differs by at most 2.
      \((b(ba)^*a + a(ab)^*b)^*\)
      Clearly this regular expression generates a subset of the language. We’ll show that if x is in the language then x is generated by the regular expression. Without loss of generality assume that no proper prefix of x has an equal number of a’s and b’s. Suppose x starts with the symbol b. Then either x=ba or x=b(ba)^k a for some k. Thus x is generated by the regular expression. A similar argument holds if x starts with the symbol a.

4. Let L_1 and L_2 be regular languages over A=\{a,b\}. Prove the following languages are regular by showing they have accepting finite automata.
   a. \(A^*-L_1\)
      Since L_1 is regular there is a DFM M such that L(M)=L_1. Assume that Q are the states of M and F are the final states. (For this construction we require that any dead states be explicit.) Consider the DFA, M’, that is identical to M except the final states of M’ are Q-F. M’ machine accepts every string in A* that is rejected by M. Thus it accepts the language A*-L_1. Since this language has a DFA it is a regular language.
   b. \(L_1 \cap L_2\)
      Let M_1 and M_2 be DFA recognizing L_1 and L_2. We construct a new machine M with states Q_1\times Q_2. From state (q_1,q_2) on symbol s we transition to state (r_1,r_2) where M_i in q_i on s transitions to r_i, i=1,2. The start state of M is (s_1, s_2) where s_i is the start state of M_i. The final states of M are (f_1, f_2) where f_i is a final state of M_i. On any input x, M concludes in state (q_1,q_2) iff M_i on x concludes in state q_i, i=1,2. Thus M accepts L_1 \cap L_2 and therefore the language must be regular.
5. Prove that any finite language is regular.

Assume $L=\{x_1, \ldots, x_n\}$. The regular expression $x_1 + x_2 + \ldots + x_n$ generates $L$ and thus $L$ is regular.