Name________________________

Homework #2

People I worked with and URL's of sites I visited:

#1. Convert to Chomsky Normal Form. Please follow the steps even if you can "see" the answer:

a) the expression grammar, G:
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow (E) \mid a$

Recursive Start

$E' \rightarrow E$
$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow (E) \mid a$

No λ productions

Chain Rules

$F \rightarrow (E) \mid a$ ok
Change $T \rightarrow T * F \mid F$ to $T \rightarrow T * F \mid (E) \mid a$
Change $E \rightarrow E + T \mid T$ to $E \rightarrow E + T \mid T * F \mid (E) \mid a$
Change $E' \rightarrow E$ to $E' \rightarrow E + T \mid T * F \mid (E) \mid a$

So have:
$E' \rightarrow E + T \mid T * F \mid (E) \mid a$
$E \rightarrow E + T \mid T * F \mid (E) \mid a$
$T \rightarrow T * F \mid (E) \mid a$
$F \rightarrow (E) \mid a$

Useless

1. **All productions produce terminal strings**
2. **All symbols reachable from S**

Chomsky Normal Form

Introduce T_a, $T_($, $T_)$, T_+, T_*:

$E' \rightarrow E\ T_+\ T$
$E' \rightarrow T\ T_*\ F$

$E' \rightarrow T_(E\ T_)$
$E' \rightarrow a$
$E \rightarrow E\ T_+\ T$
$E \rightarrow T\ T_*\ F$
$E \rightarrow T_(E\ T_)$
$E \rightarrow a$
$T \rightarrow T\ T_*\ F$
$T \rightarrow T_(E\ T_)$
$T \rightarrow a$
$F \rightarrow T_(E\ T_)$
$F \rightarrow a$
$T_a \rightarrow a$
$T_(\rightarrow ($
$T_) \rightarrow)$
$T_+ \rightarrow +$
$T_* \rightarrow *$

Introduce Intermediate variables: V_1, V_2, V_3, V_4, V_5:

$E' \rightarrow T\ V_1$
$V_1 \rightarrow E\ T_)$
$E' \rightarrow a$
$E \rightarrow E\ V_2$
$V_2 \rightarrow T_+\ T$
$E \rightarrow T\ V_3$
$V_3 \rightarrow T_*\ F$
$T \rightarrow T_(V_4$
$E \rightarrow a$
$V_4 \rightarrow E\ T_)$
$T \rightarrow a$
$F \rightarrow T_(V_5$
$V_5 \rightarrow E\ T_)$
$F \rightarrow a$
$T_a \rightarrow a$
$T_(\rightarrow ($
$T_) \rightarrow)$
$T_+ \rightarrow +$
$T_* \rightarrow *$

b) $S \rightarrow A \mid A\ B\ a \mid A\ b\ A$
$A \rightarrow A\ a \mid \lambda$
$B \rightarrow B\ b \mid B\ C$
$C \rightarrow C\ B \mid C\ A \mid b\ B$

Recursive Start

none

Remove λ Productions

Null = {A, S}

$C \rightarrow C\ B \mid C\ A \mid b\ B$
$B \rightarrow B\ b \mid B\ C$
$A \rightarrow A\ a \mid a$
$S \rightarrow A \mid A\ B\ a \mid A\ b\ A \mid B\ a \mid b\ A \mid A\ b \mid b \mid \lambda$

or
$S \rightarrow A \mid A\ B\ a \mid A\ b\ A \mid B\ a \mid b\ A \mid A\ b \mid b \mid \lambda$
$A \rightarrow A\ a \mid a$
$B \rightarrow B\ b \mid B\ C$
$C \rightarrow C\ B \mid C\ A \mid b\ B$

Remove chain rules

$S \rightarrow A\ a \mid a \mid A\ B\ a \mid A\ b\ A \mid B\ a \mid b\ A \mid A\ b \mid b \mid \lambda$
$A \rightarrow A\ a \mid a$
$B \rightarrow B\ b \mid B\ C$
$C \rightarrow C\ B \mid C\ A \mid b\ B$

Remove useless

Term = {A, S}

so have:

$S \rightarrow A\ a \mid a \mid A\ b\ A \mid b\ A \mid A\ b \mid b \mid \lambda$
$A \rightarrow A\ a \mid a$

Reach = {S, A}
so above grammar is ok.

Chomsky Normal Form

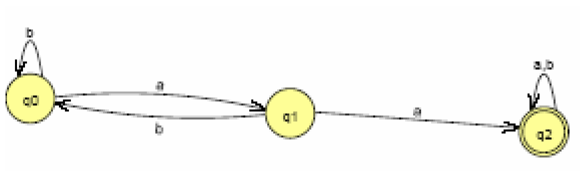
Introduce new variables: T_a, T_b
$S \rightarrow A\ T_a \mid a \mid A\ T_b\ A \mid T_b\ A \mid A\ T_b \mid b \mid \lambda$
$A \rightarrow A\ T_a \mid a$
$T_a \rightarrow a$
$T_b \rightarrow b$

Introduce new variables: V_1
$S \rightarrow A\ T_a \mid a \mid A\ V_1 \mid T_b\ A \mid A\ T_b \mid b \mid \lambda$

$V_1 \rightarrow T_b\ A$
$A \rightarrow A\ T_a \mid a$
$T_a \rightarrow a$
$T_b \rightarrow b$

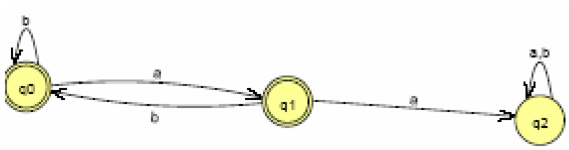
#2. Show the following languages are regular by creating finite automata with L = L(M)

a) Strings over {a,b} that contain 2 consecutive *a*'s



	a	**b**
>q_0	**q_1**	**q_0**
q_1	**q_2**	**q_0**
***q_2**	**q_2**	**q_2**

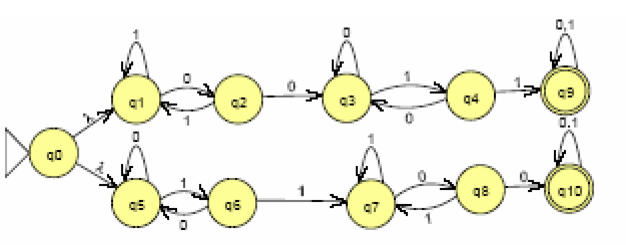
b) Strings over {a,b} that do not contain 2 consecutive *a*'s



	a	**b**
>*q_0	**q_1**	**q_0**
***q_1**	**q_2**	**q_0**
q_2	**q_2**	**q_2**

c) The set of strings over {0,1} which contain the substring *00* and the substring *11*

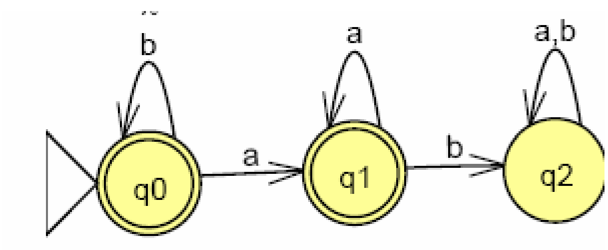
Problem doesn't say whether this must be a dfa and this is easier with an nfa:



	λ	0	1
>q_0	q_1 , q_5		
q_1		q_2	
q_2		q_3	q_1
q_3		q_3	q_4
q_4		q_3	q_9
q_5		q_5	q_6
q_6		q_5	q_7
q_7		q_8	q_7
q_8		q_{10}	q_7
*q_9		q_9	q_9
*q_{10}		q_{10}	q_{10}

d) The set of strings over {a,b} which do not contain the substring *ab.*

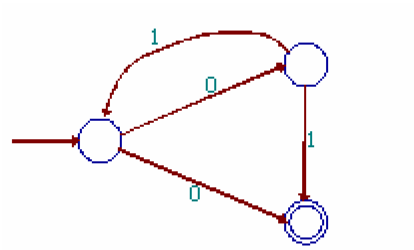
Similar to parts a and b, I will first create a fa that does accept *a b* and then I will reverse the final and the nonfinal states:



	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

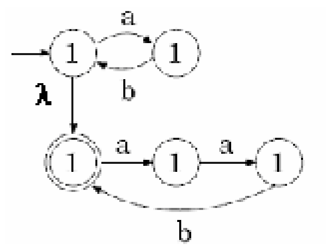
#3. Describe L(M) for the following nfa's: a) in words and b) as a regular expression

a)



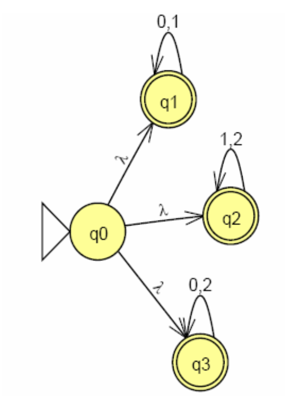
L(M) = Alternating 0's and 1's (including none) that begin with a 0
(01)* (01 U 0)

b)



0 or more *ab*'s followed optionally by 0 or more *aab*'s
(ab)* (aab)*

#4. a) Create an NFA (with λ transitions) for all strings over {0, 1, 2} that are missing at least one symbol. For example, *00010*, *1221*,and *222* are all in L while *221012* is not in L.



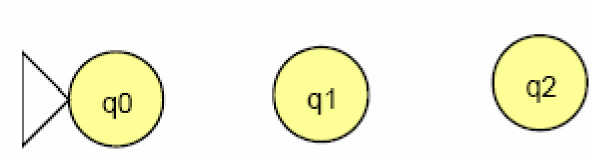
b) Given an NFA with several final states, show how to convert it into one with exactly one start state and exactly one final state.

Create a new initial state and a λ-transition from it to all the original start states

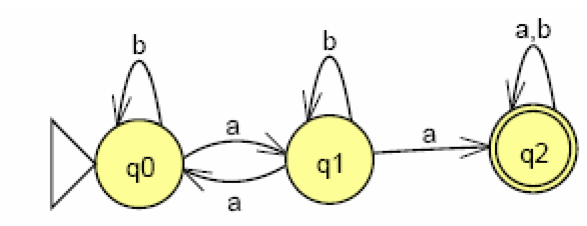
Create a new final state and a λ-transition from all the original final states (which mark to no longer be final) to this new final state

c) Suppose an NFA with k states accepts at least one string. Show that it accepts a string of length k-1 or less.

Look at a fa with 3 states:



No matter how you draw the transitions or which states are final states, to process a string of length *k* means you visited a state twice. For example:



accepts the string of length 3: aba

But just by not visiting the revisited state (q_1), this will accept a a (of length 2)

In general, if a string of length k is accepted by a fa with k states, it visits (at least) 1 state twice. By not visiting this state the 2nd time (e.g., don't take the loop), we can accept a string with 1 fewer symbol, i.e, of length k – 1.

d) Let L be a regular language. Show that the language consisting of all strings not in L is also regular.
If L is regular, there is a dfa, M, such that L = L(M), that is, M accepts L. If we create a new finite automaton, M', by reversing final and non-final states, we will accept what M didn't and reject what M accepted; that is, C(L) = L(M')

#5. a) Consider the extended transition function, δ*, defined by:

$$\delta^*(q,\lambda) = q$$
$$\delta^*(q,wa) = \delta(\delta^*(q,w),a)$$

a) Show that $\delta^*(q,a) = \delta(q,a)$ (follows from the definition)

$$\boldsymbol{\delta^*(q,a) = \delta(\delta^*(q,\lambda),a) = \delta(q,a)}$$

b) Show that $\delta^*(q, uv) = \delta^*(\delta^*(q,u),v)$ (use induction)

Proof by induction on $|v|$

Basis When $|v| = 0$, $v = \lambda$, and
left-hand-side: $\delta^*(q, u\lambda) = \delta^*(q, u)$ (Property of λ)
right-hand-side: $\boldsymbol{\delta^*(\delta^*(q,u), \lambda)} = \delta^*(q, u)$ (Definition of δ^*)

Induction Hypothesis

$\delta^*(q, uv) = \delta^*(\delta^*(q,u),v)$ for $0 \leq |v| \leq n$

Induction Step: To show $\delta^*(q, uv) = \delta^*(\delta^*(q,u),v)$ for $|v| = n + 1$:

Since $|v| = n + 1$, and $n \geq 0$, v an be written *wa* where $|w| = n$ and $a \varepsilon \Sigma^*$

left-hand-side: $\delta^*(q, uv) = \delta^*(q, u(wa))$ **substituting wa for v**
$= \delta^*(q, (uw)a)$ **associativity of concatenation**
$= \delta(\delta^*(q, uw), a)$ **definition of** $\boldsymbol{\delta^*}$
= $\delta(\delta^*(\delta^*(q,u),w),a)$ **IH**
= $\delta^*(\delta^*(q,u),wa)$ **definition of** $\boldsymbol{\delta^*}$
= $\delta^*(\delta^*(q,u)\ v)$ **v = wa**
= right-hand-side

c) Show that $\delta^*(q,aw) = \delta^*(\delta(q,a),w)$ (follows from above)

Considering symbol “a” as a string:

$\delta^*(q, aw) = \delta^*(\delta^*(q,a),w)$ **by part b**
$= \delta^*(\delta(q,a),w)$ **by part a**