(1) Using *TopHat*, the "Worksheet" titled *S21 HW2 Online Part.* There are 10 questions, each worth 2 points, for 20 total. *Answers given there.*

(2) For each of the following languages A, write a regular expression r such that $L(r) = A$, and then give an NFA N_r such that $L(N_r) = A$. Well, if you give a DFA, that counts as an NFA, but in one or two cases you may find the NFA easier to build especially once you have r. For part (b), note that a string can be broken uniquely into maximal "blocks" of consecutive letters. For instance, in "Tennessee" the blocks are T, e, nn, e again, ss, and ee.

(a) The language of strings over $\{a, b\}$ in which every b is followed immediately by at least two a's.

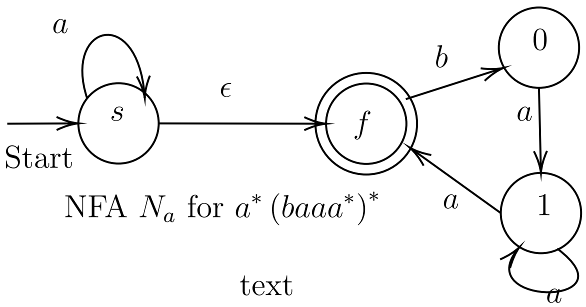
(b) The language of strings over $\{a, b\}$ in which every a belongs to a "block" of at least 2 a's and every b belongs to a block of at least 3 b's.

(c) The language of strings over $\{a, b\}$ with at least 3 characters, such that the last character equals the third-from-last character. (6 + 6 + 12 = 24 pts.)

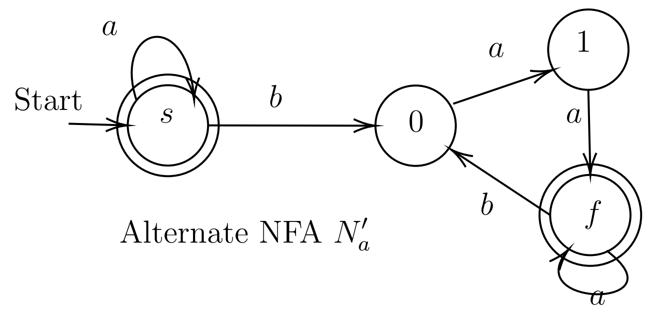
Answer: (a) The attempt $a^*(baa)^*$ gives the strings in which every b is followed by exactly two a's. To get "at least two a's" it must be $a^*(baaa^*)^*$ or $a^*(baa^+)^*$. Note that this allows strings of only a's, including the empty string, for which the condition "in which every $b\dots$" holds true by default. The NFA N_a shown below transcribes this expression; it can also be written without the ϵ-arc by using b to move off the start state, shown as N'_a.

(b) This is $(aaa^* \cup bbbb^*)^*$. If you like superscript-plus, you can write this as $(aa^+ \cup bbb^+)^*$. Note again that the empty string is allowed since with no blocks at all the condition holds by default. The NFA N_b adds two loops to what you would use for $(aa \cup bbb)^*$.

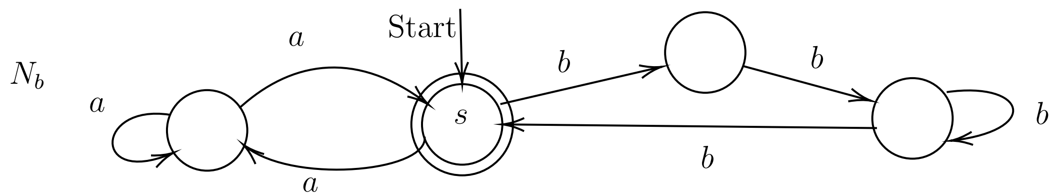
(c) Here there must be at least three characters—indeed, that would be the interpretation even if the prose had omitted the clause "with at least 3 characters," since "the third-from-last character" is a positive mention. The "multiplied-out" regular expression is $(a + b)^*(aaa + aba + bab + bbb)$. You can also "factor" it as $(a + b)^*(a(a + b)a + b(a + b)b)$. (Why did I switch to writing + rather than $\cup$? To help visualize the factoring.) The NFA N_c starts by imitating the design of the NFA for the "third-last-char-equals-1" language but has a branch at that point. [The question did not ask to convert it to a DFA, but this is shown as a bonus example.]



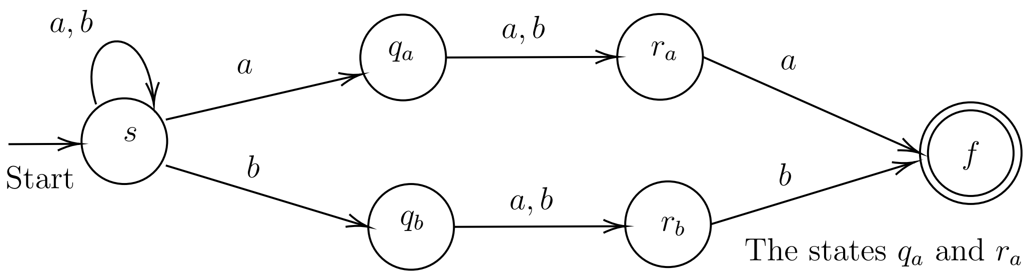
NFA N_a for $a^*(baaa^*)^*$



Alternate NFA N_a'



NFA N_b for $(aaa^* + bbbb^*)^*$



The states q_a and r_a "remember" that an a was read at the fork out of state s, while q_b, r_b remember b.

NFA N_c for $(a+b)^*(a(a+b)a+b(a+b)b)$

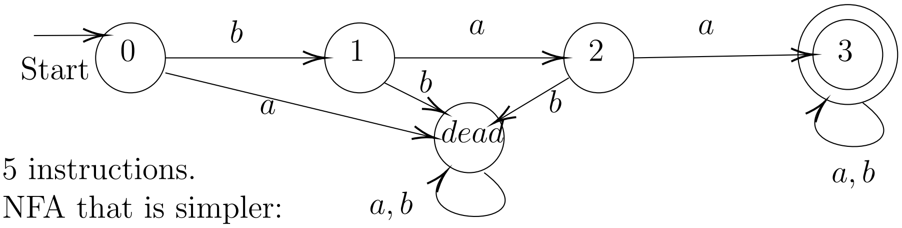
(3) (a) Again over $\Sigma = \{a, b\}$, design a DFA M such that $L(M)$ equals the language of strings that begin with baa. Note that if you delete the dead state and the edges involving it, you get what is technically an NFA with only 5 instructions.

(b) Now design an NFA N with only 5 instructions such that $L(N)$ equals the language of strings that end in aab. (As in part (a), a single edge or loop labeled with two chars counts as two instructions.)

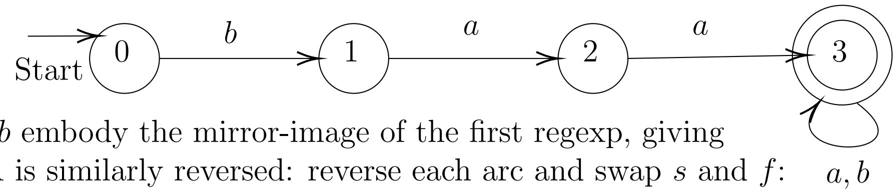
(c) Then show the conversion of N into an equivalent DFA, following the method in class. Compare the number of instructions and states that you get between the two. (6+6+12 = 24 pts., for 68 total on the set)

Answers all in the picture:

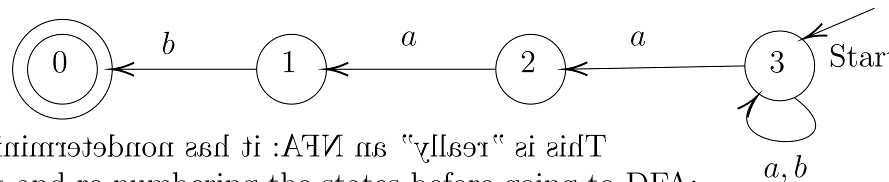
DFA for $baa(a \cup b)^*$:



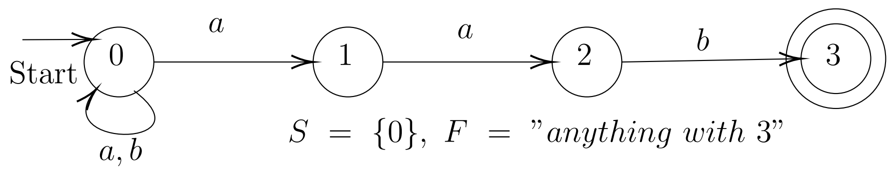
The dead state hogs 5 instructions.
Deleting it yields an NFA that is simpler:



Strings that end in aab embody the mirror-image of the first regexp, giving $(a \cup b)^* aab$. The NFA is similarly reversed: reverse each arc and swap s and f:



This is "really" an NFA: it has nondeterminism on a at the start state.
OK, writing it left-to-right and re-numbering the states before going to DFA:



$S = \{0\}$, $F =$ "*anything with* 3"

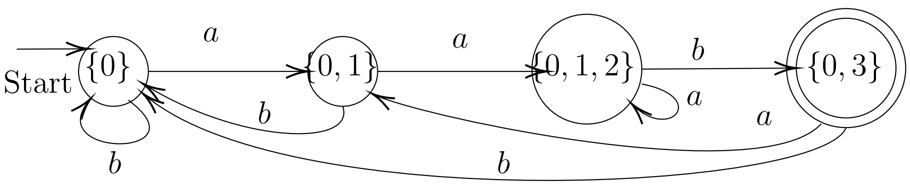
$\underline{\delta}$	a	b
0	$\{0,1\}$	$\{0\}$
1	$\{2\}$	$\emptyset$
2	$\emptyset$	$\{3\}$
3	$\emptyset$	$\emptyset$

$\Delta(\{0\}, a) = \{0,1\}$, $\Delta(\{0\}, b) = \{0\}$

$\Delta(\{0,1\}, a) = \{0,1,2\}$, $\Delta(\{0,1\}, b) = \{0\}$

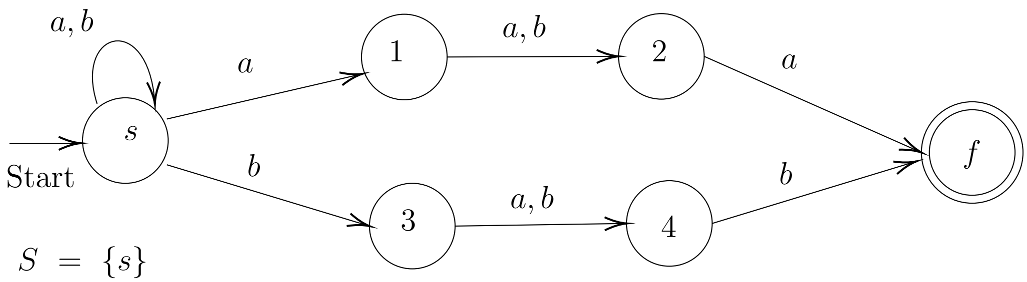
$\Delta(\{0,1,2\}, a) = \{0,1,2\}$, $\Delta(\{0,1,2\}, b) = \{0,3\}$

$\Delta(\{0,3\}, a) = \{0,1\}$, $\Delta(\{0,3\}, b) = \{0\}$. *Done.*



The DFA has the same number of states but 8 instructions instead of 5. Its "backbone" is the same, however, and the "design pattern" of tracking progress toward the goal would design it directly.

Extra—this was not assigned but is a useful example. Here is the NFA-to-DFA conversion for 2(c):



$\underline{\delta}$	a	b
s	$\{s,1\}$	$\{s,3\}$
1	$\{2\}$	$\{2\}$
2	$\{f\}$	$\emptyset$
3	$\{4\}$	$\{4\}$
4	$\emptyset$	$\{f\}$
f	$\emptyset$	$\emptyset$

$S = \{s\}$

$\Delta(S,a) = \{s,1\}$, $\Delta(S,b) = \{s,3\}$

$\Delta(\{s,1\},a) = \{s,1,2\}$, $\Delta(\{s,1\},b) = \{s,2,3\}$

$\Delta(\{s,3\},a) = \{s,1,4\}$, $\Delta(\{s,3\},b) = \{s,3,4\}$

Ouch, we have four new states to expand. How bad will this get?

$\Delta(\{s,1,2\},a) = \{s,1,2,f\}$, $\Delta(\{s,1,2\},b) = \{s,2,3\}$

$\Delta(\{s,2,3\},a) = \{s,1,4,f\}$, $\Delta(\{s,2,3\},b) = \{s,3,4\}$

$\Delta(\{s,1,4\},a) = \{s,1,2\}$, $\Delta(\{s,1,4\},b) = \{s,2,3,f\}$

$\Delta(\{s,3,4\},a) = \{s,1,4\}$, $\Delta(\{s,3,4\},b) = \{s,3,4,f\}$

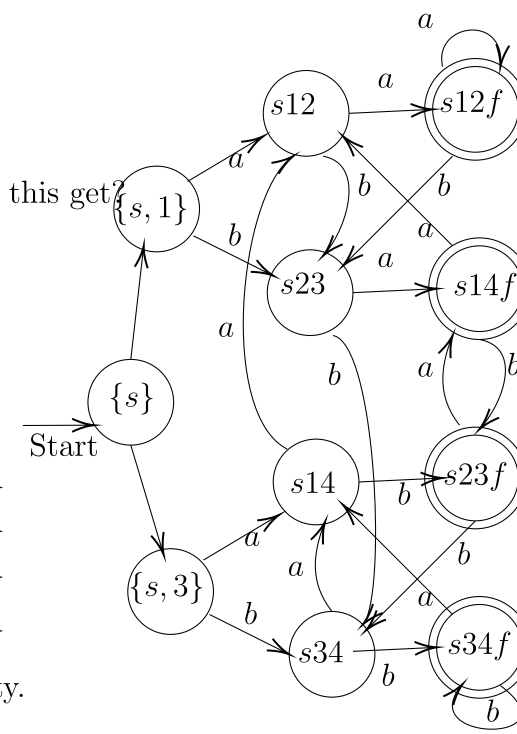
Answer was worse—four new states again. The end?

$\Delta(\{s,1,2,f\},a) = \{s,1,2,f\}$, $\Delta(\{s,1,2,f\},b) = \{s,2,3\}$

$\Delta(\{s,2,3,f\},a) = \{s,1,4,f\}$, $\Delta(\{s,2,3,f\},b) = \{s,3,4\}$

$\Delta(\{s,1,4,f\},a) = \{s,1,2\}$, $\Delta(\{s,1,4,f\},b) = \{s,2,3,f\}$

$\Delta(\{s,3,4,f\},a) = \{s,1,4\}$, $\Delta(\{s,3,4,f\},b) = \{s,3,4,f\}$



DFA M at right. Like binary tree at first but gets twisty.