COSC 362, Test #1A
Solutions

Note: On test #1B, problems #1, 2, and 4 are equivalent to the corresponding problems on test #1A; just switch all a's and b's in each. Problems #3, 5, and 6 are identical on both tests.

1. a) Find a grammar that generates the set of all strings $a^n b^m$ such that n is odd and m is even.

   Solution: One such grammar is as follows:

$$S \to AB$$
$$A \to aaA \mid a$$
$$B \to bbB \mid \lambda$$

   In this grammar, k iterations of $A \to aaA$ followed by $A \to a$ will yield $A \overset{*}{\to} a^{2k}A \to a^{2k+1}$.

   Similarly, j iterations of $B \to bbB$ followed by $B \to \lambda$ will yield $B \overset{*}{\to} b^{2j}B \to b^{2j}$

   Thus, for any natural numbers k, j, we can generate $S \to AB \overset{*}{\to} a^{2k+1}b^{2j}$. This shows that our grammar generates the set of all strings $a^n b^m$ such that n is odd and m is even.

   Comment: Another, simpler example some of you found is as follows:

$$S \to aA$$
$$A \to aaA \mid Abb \mid \lambda$$

   This grammar is also valid, with the obvious advantage that it requires one less variable.

   b) Give a simple description (one sentence should be enough) of the language generated by the grammar with start variable S and productions

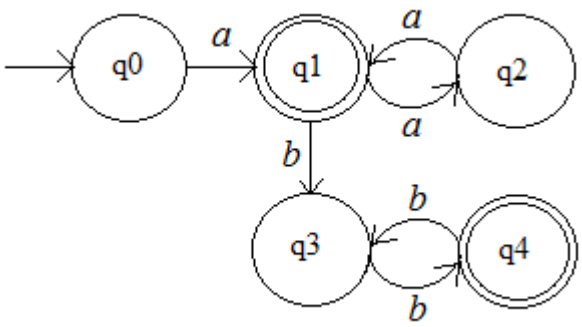
$$S \to AbAbA$$
$$A \to aA$$
$$A \to \lambda$$

   Solution: This grammar generates all strings in Σ^* that contain exactly two b's. (There are other valid descriptions, but this is the simplest one.)

   Comment: The answer "strings of the form $a^n ba^n ba^n, n \geq 0$" is technically incorrect, since the strings of a's may be of different length. Writing "n" for all three implies that all three strings must be the same length. A similar, but valid, description would be "strings of the form $a^m ba^n ba^p$, where $m \geq 0, n \geq 0, p \geq 0$."

2. a) Design a NFA that accepts the set of all strings $a^n b^m$ where n is odd and m is even. Do not use any "trap states" in your NFA.

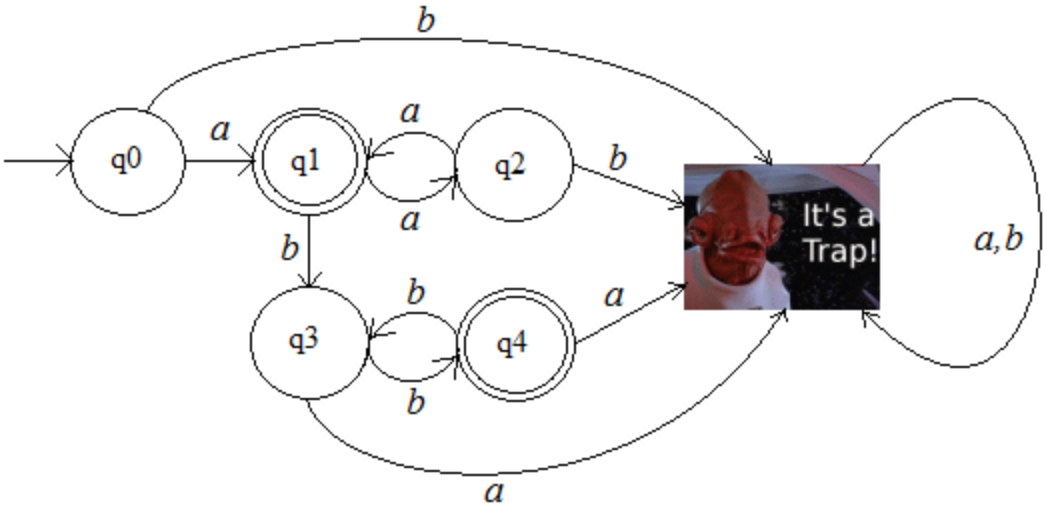
Solution: One such NFA is as follows.



Note that there are other valid solutions. For example, we could redesign the above NFA as follows: replace $\delta(q1, b) = q3$ with $\delta(q1, \lambda) = q3$, make q3 a final state, and make q4 a nonfinal state. The resulting NFA would accept the same language. (However, designing the NFA without λ –transitions makes conversion to a DFA easier…)

b) Design a DFA that accepts the set of all strings $a^n b^m$ where n is odd and m is even. (Note that a DFA for this language will require a "trap state.")

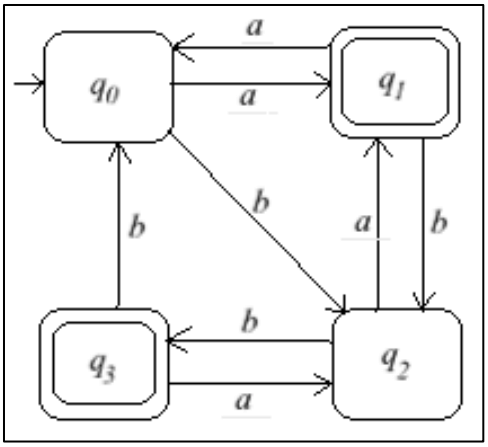
Solution: The NFA shown above is easily converted to a DFA by simply adding all of the "missing" arrows and connecting them to a trap state. (The trap state in the DFA is equivalent to getting "stuck" in the NFA.) This gives us the following DFA:



3. Consider the DFA shown below, with initial state q_0 and final states q_1, q_3.

Answer each of the following questions about the language accepted by this DFA.

a. Determine whether each of the following strings is accepted: $aaa, bbb, aaab, bbba$. (No work is necessary; just state which string(s) is/are accepted.)



Answers:
$\delta^*(q_0, aaa) = q_1$, so aaa is accepted.
$\delta^*(q_0, bbb) = q_0$, so bbb is not accepted.
$\delta^*(q_0, aaab) = q_2$, so $aaab$ is not accepted.
$\delta^*(q_0, bbba) = q_1$, so $bbba$ is accepted.

b. For what values of n is the string a^n accepted? Explain your answer.

Solution: We can see from the diagram that the computation on a strings of a's will end in state q_0 if there is an even number of a's, and in state q_1 if there is an odd number of a's. Therefore, a^n is accepted if and only if $\boldsymbol{n}$ **is an odd number**.

c. For what values of m is the string b^m accepted? Explain your answer.

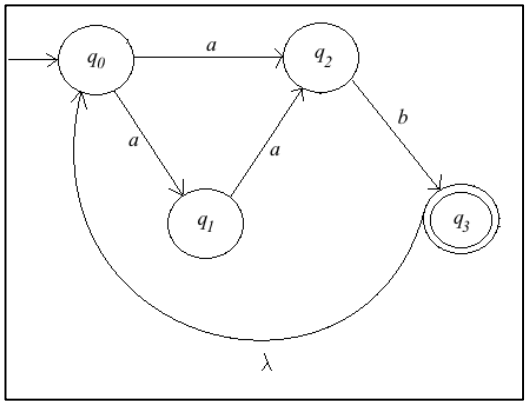
Solution: The computation on a string of b's cycles through q2, then q3, then q0. We can see that $\delta^*(q_0, bb) = q_3$, and that $\delta^*(q_3, bbb) = q_3$; that is, after the first two b's, we cycle back to q3 every third b after that. So, the strings of b's whose computations end in the final state q_3 are $bb, bbbbb, b^8, b^{11}, b^{14}$, … In general, b^m is accepted if and only if $\boldsymbol{m}$ **is 2 more than a multiple of 3.** (Or, equivalently, $m \equiv 2\ (mod\ 3)$.)

EXTRA CREDIT: (+5 points for the first person who turns in a correct PROOF of the following)

Prove that this DFA does not accept any string with suffix aab, but it accepts all strings with suffix $aaba$ or $aabb$.

4. a) Convert the NFA shown below into an equivalent DFA.

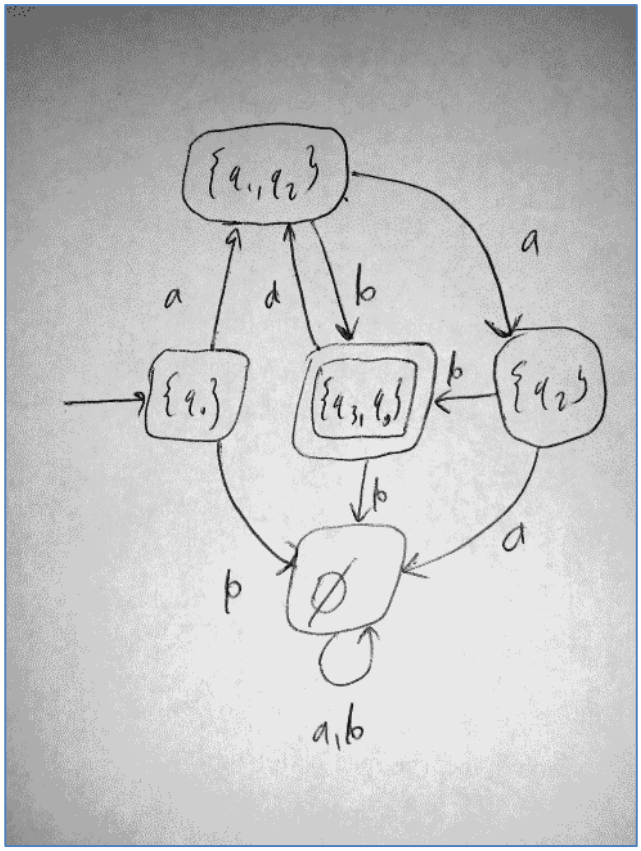
Solution: Following the conversion procedure covered in class, you should end up with something equivalent to the DFA shown in the photo below.



b) Briefly describe the language accepted by this NFA – or, equivalently, by the DFA you found in part (a). (One or two sentences should be sufficient.)

Note: there are several valid descriptions of this language. For example…

Answer: The NFA (or DFA) accepts all strings that start with an *a*, end with a *b*, and do not have either *bb* or *aaa* as a substring.



Equivalent answer:
The NFA consists of all strings that start with *a* and end with *b* such that every *b* (except the last one) is followed by an *a*, and there are never more than two *a*'s in a row.

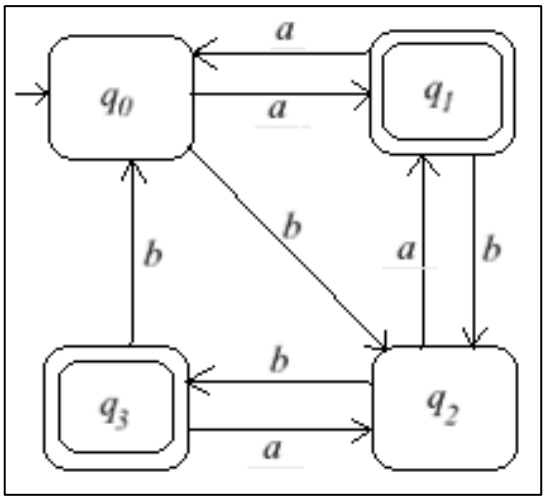
Example: (set notation)
The NFA accepts the "positive closure" of $\{ab, aab\}$ – that is, the set of concatenations of at least one string from the set $\{ab, aab\}$ (with repetition allowed). (Or, simply, $L = \{ab, aab\}^+$).

(Note: recall "positive closure" is the same as "star closure" except that it excludes the empty string. How could we modify the NFA in this problem to accept $\{ab, aab\}^*$?)

5. Show that the DFA below is minimal. (That is, show that no two states in the DFA are indistinguishable.)

Solution: Start by noting that we have $F = \{q_1, q_3\}$ and $F' = \{q_0, q_2\}$.

Next, we'll see if either of these pairs of states is distinguishable by strings of length 1 (that is, a or b):



$\delta(q_1, a) = q_0 \notin F \quad \delta(q_3, a) = q_2 \notin F$
$\delta(q_1, b) = q_2 \notin F \quad \delta(q_3, b) = q_0 \notin F$

There is no distinction between q_1 and q_3 here, so these two states are not distinguishable by any string of length 1.

On the other hand, with q_0 and q_2 we get the following result:

$\delta(q_0, a) = q_1 \in F \quad \delta(q_2, a) = q_1 \in F$
$\boldsymbol{\delta(q_0, b) = q_2 \notin F} \quad \boldsymbol{\delta(q_2, b) = q_3 \in F}$

The point here is that q_0 and q_2 are distinguished by b.

To show that q_1 is distinguishable by q_3, we must look at strings of length at least 2. It turns out that these two states are distinguished by the string ab:

$$\delta^*(q_1, ab) = \delta(q_0, b) = q_2 \notin F$$
$$\delta^*(q_3, ab) = \delta(q_2, b) = q_3 \in F$$

(Note: the string *bb* also works here.)

Since the computation on ab ends in a final state starting from q_1 but ain a non-final state starting from q_3, we conclude that q_1 and q_3 are distinguishable.

Thus, every pair of states in the DFA is distinguished by some string in Σ^*, which means the DFA is minimal.

6. Prove by induction: for all positive integers n,

$$\sum_{i=1}^{n} 2i = n^2 + n.$$

(Reminder: $\sum_{i=1}^{n} 2i = 2 + 4 + \cdots + 2n$; that is, the sum of the first n even positive integers.)

Proof (by induction):

Base (or initial) step: When $n = 1$, the given equation is 2=1+1, which is obviously true.

Inductive Hypothesis: Assume $\sum_{i=1}^{n} 2i = n^2 + n$ is true for all positive integers $1 \leq n \leq k$. In particular, assume the equation is true when $n = k$; that is, we assume

$$\sum_{i=1}^{k} 2i = k^2 + k.$$

It follows that

$$\begin{aligned}\sum_{i=1}^{k+1} 2i &= \sum_{i=1}^{k} 2i + 2(k+1) \\ &= (k^2 + k) + 2(k+1) \\ &= k^2 + 3k + 2 \\ &= (k^2 + 2k + 1) + (k+1) \\ &= (k+1)^2 + (k+1),\end{aligned}$$

which is exactly the given equation when $n = k + 1$.

Thus, for any positive integer k, the equation is true for $n = k + 1$ if it is true for $n = k$.

Therefore, by induction, the equation is true for all positive integers n.