The algorithm discussed in class for constructing a machine M to accept $L_1 \cap L_2$ given machines M_1 and M_2 for L_1 and L_2 respectively is for DFAs. The two NFAs given in #1 must first be converted to DFAs. (It is also possible to do the conversion at the same time as constructing M, or to adapt the intersection construction to work with NFAs instead of DFAs. Both of these were also accepted.) Also, remember that when there isn't a transition shown, it means that it is a transition to a trap state and that string won't be accepted — either add a trap state and the requisite transitions to each DFA before doing the intersection or only include transitions in M for which there are transitions in both M_1 and M_2. Do not treat a missing transition as if it is a transition that loops back to the same state.

$a^n b^m$ means n as followed by m bs. n as and m bs in any order would be
$\{\ w \in \{a,b\}^* \mid n_a(w) = n \wedge n_b(w) = m\ \}$.

When constructing a grammar for a language, remember that the grammar needs to generate *all* of the strings in the language, not just some of them. For example, #4a asks for a grammar to generate strings of the form $a^n b^m$ where $n \neq m$. It isn't sufficient to create a grammar than can generate strings with one more a (or b) — those strings are part of the language, but so are many other things. The grammar needs to be able to generate every string of the required language.

When counts are related to each other, symbols need to be generated in pairs. The following does *not* work for #4b:

$$\begin{aligned}
S &\longrightarrow ABC \\
A &\longrightarrow aA \\
A &\longrightarrow \epsilon \\
B &\longrightarrow bB \\
B &\longrightarrow \epsilon \\
C &\longrightarrow cC \\
C &\longrightarrow \epsilon
\end{aligned}$$

Any number of as, bs, and cs can be generated, with the as followed by bs and the bs followed by cs, but there's nothing to enforce that the number of bs is greater than the combined number of as and cs. To ensure more bs, first ensure the same number of bs — every a or c produced must also result in a b — and then allow one or more additional bs.