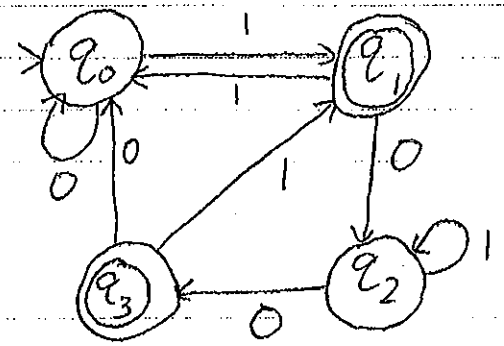
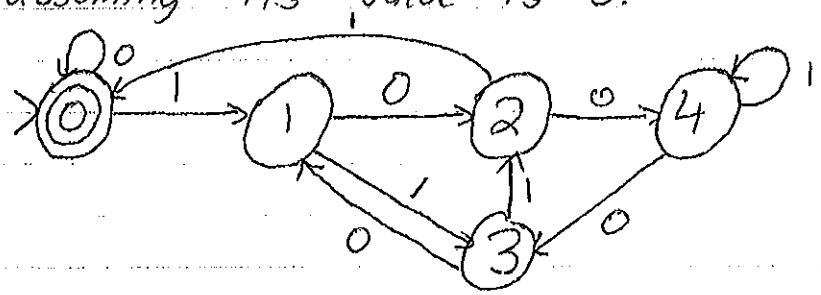
COT 4210 HMK #1 SOLUTIONS (SUM '12)

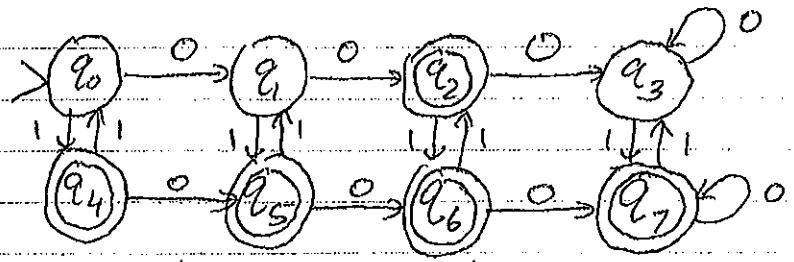
1)



2) States are 0, 1, 2, 3, 4, representing 0 mod 5, 1 mod 5, 2 mod 5, 3 mod 5 and 4 mod 5. $\Sigma = \{0, 1\}$. In this solution, we'll accept ε, assuming its value is 0.

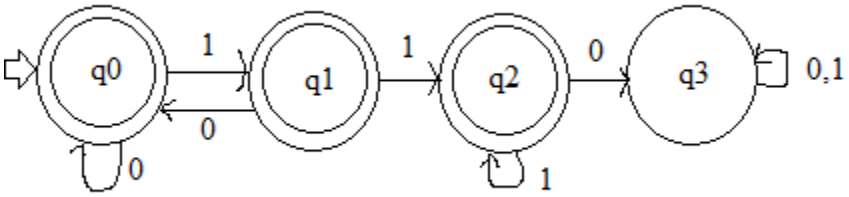


3)

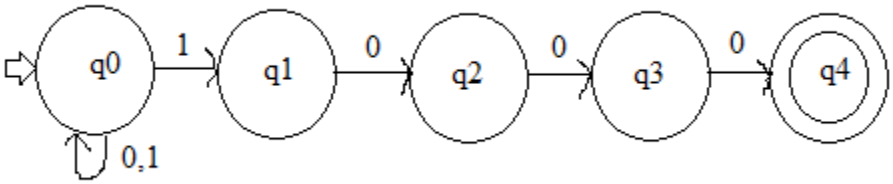


The basic idea here is that the top row represents an even # of 1s and the bottom row represents an odd # of 1s. The state number mod 4 represents how many zeroes the string has, with states q_3, q_7 representing "3 or more 0s".

4) The general strategy in the DFA below is to keep track of how far along the end of the current string is in forming 110. The start state, q0, indicates that none of the three characters are last. The next state, q1, indicates that we've read in 1 as the last character, but not 11 as the last two. The state q2 indicates that the last two characters read in are 11 and the state q3 means that we've seen 110 as a substring before. If our last character is a 1 and then we read in a 0, we've destroyed our potential formation of 110 and go back to q0. If our last character is 1 and we read in a 1, then our last two characters are 11. Similarly, if our last two characters are 11 and we read in a 1, then still, we haven't matched 110 but our last two characters are still 11. Here is the DFA:



5) Since we are designing an NFA here, the description of the language simply means that we accept all strings that end in 1000. Thus, we can build a single path for 1000 across five states that only transition on the correct characters, and then loop 0,1 at the start, essentially letting our machine "guess" when we've gotten to the last four characters of the string:



6) Though the description is obfuscated, once you think through what it means to have exactly one substring 10 and 01, you realize that this really means strings that start and end with the same character and only have one run of the opposite character in between. In an adapted form of a regular expression, the desired language is $1^+0^+1^+ \cup 0^+1^+0^+$. To build our NFA, we can create two separate branches, one for the first character 1 and the other for the first character 0. Then, we move to separate states each time a different character (transition either 01 or 10) occurs: