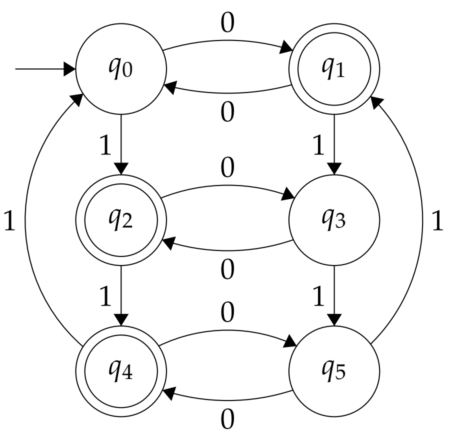
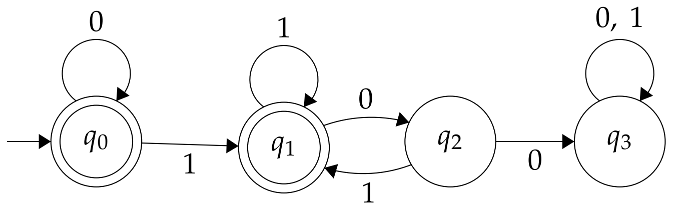
Problem 1



My DFA is the cross product of a DFA that accepts the number of 0's that is even, and a DFA that accepts the number of 1's that is not divisible by 3.

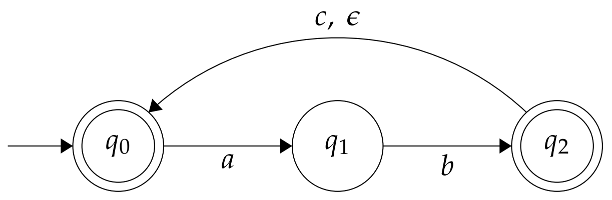
q_1, q_2, q_4 are the accept states because the condition 'if and only if' is only true when the antecedent and consequent are both true, or both false. Therefore q_2 and q_4 are accept states because the number of 0's is even and the number of 1's is not divisible by 3. And q_1 is an accept state because the number of 0's is odd and the number of 1's is divisible by 3. Therefore my DFA is correct as it will accept the language all strings in which the number of 0's is even if and only if the number of 1's is not divisible by 3.

Problem 2



The DFA I have constructed will accept the language because any number of leading 0's in the string will be accepted in q_0. Then if any number of 1's follow it will be accepted in q_1. q_2 and q_3 are designed to reject strings not in this language. It does so by making sure there is always a 1 after a 0, and also if there is not a second 0 after a 1.

Problem 3



My NFA construction has three 3 states. The NFA checks for if there is a 'ab', then it will start a new thread, accepting anything that follows that is not a 'b', thus it will reject strings like abb. If it sees a 'a' after 'ab' the new thread will effectively allow it to go into q_1 which starts the cycle again. Therefore this NFA can accept any combination of 'ab' and 'abc' while rejecting those strings that are not a part of this language.

Problem 4

1. Here is the solution for part 1.

   L_2 will accept strings where the difference of the number of a's and b's is no greater than 2. So when you concatenate L_b with it, the new language will contain at least 4 b's. The intersection between this and L_2 will have to contain the strings with at least 4 b's, so then there must be so there must be at least 2 a's as well to be a part of L_2. Then lastly it is concatenated with L_a which adds two more a's to it so now it will have at least four a's with a equal number of b's.

   $L = L_a(L_2L_b \cap L_2) = \{a^ib^j \mid i \geq 4\,,\ j = i\}$

2. Here is the solution for part 2.
   This NFA will recognize strings that contain two 0's separated by a sub-string with a length that is a multiple of 3. The regular expression for this language is (0+1)*0((0+1)(0+1)(0+1))*0(0+1)*.

   The expression is correct because the expression starts and ends with (0+1)*0...*0(0+1)*, this part will require there be at least two 0's. The inside part ((0+1)(0+1)(0+1))* will require you to choose either a 0 or 1 three times, and since it is closed under *, the length can be any multiple of 3. Therefore this expression will recognize strings that contain two 0's separated by a sub-string with a length that is a multiple of 3.