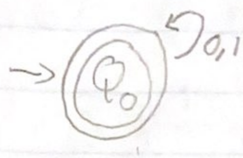
CS 3133 Hw 1

1. A language is regular if it is the language of a DFA and can be represented by a regular expression.



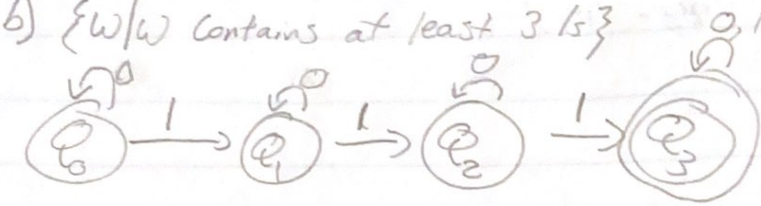
Explanation: This DFA accepts the empty string as well as (all binary strings) any combination of 0, 1, for this is a self referential accepting state. Because we can represent as a DFA, this language is regular.

2. $\Sigma^* S_0 \Sigma^* S_1 \Sigma^* S_2 \ldots \Sigma^* S_n$

Explanation: In this proof, Σ^* is used to represent any length and combinations of binary strings proceeding $S_0, S_1, S_2, \ldots S_n$. This means, if we only care that there exists a substring $S_0, S_1, \ldots S_n$ in the full length of the string and can be in any position, then it does not matter where $S_0, S_1, \ldots S_n$ are positioned in the string, so long as they exist in order.

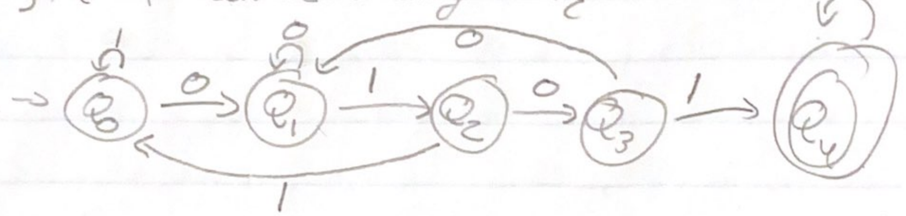
3) Exercise 1.6 from book b, c, d

b) {w | w contains at least 3 1s}



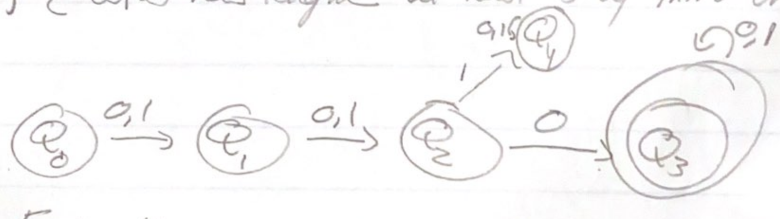
Explanation: This is a DFA (b/c for every possible input, there is exactly 1 path for each string) and this DFA is guaranteed to output at least 3 1's in order to reach the accepting state Q_3, otherwise it will cycle through 0's. If it doesn't find 3 1's, it does not accept.

c) {w | w contains substring 0101}



Explanation: Q_0 begins with a self referential loop that is used to filter out any binary strings that do not contain "0101". $Q_{0,1,2,3}$ are positioned such that to reach Q_4, you must have "0101" but can still accept if you have more binary strings after "0101". The loops on Q_0, Q_1, & Q_2 are also positioned such that if there exists a char that is not in the order "0101", then it goes back to the self referential loops to continue filtering the strings. For example, Q_1 acts as the checker for if the string starts w/ 0. This DFA does not accept/does not reach the accepting state Q_4 if there does not exist "0101"

d) {w | w has length at least 3 w/ third character 0}



Explanation: this is a DFA b/c for every possible input, there is exactly 1 path for each string and if this DFA does not have a 0 in it's third position, accounting for all binary strings, then it will go to Q_4 and not accept, otherwise it will go to Q_3 and accept.

4) Exercise 1.20 b, c, d, e

b)	Are members	Are not members
	1. abababab	1. bbababa
	2. abab	2. baba

Explanation: this expression accepts any combination of abab.... where a needs to be followed by a then b, not b then a.

c)	Are members	Are not members
	1. aa	1. abab
	2. bb	2. bbaa

Explanation: This expression accepts any # of a or b, not both a & b.

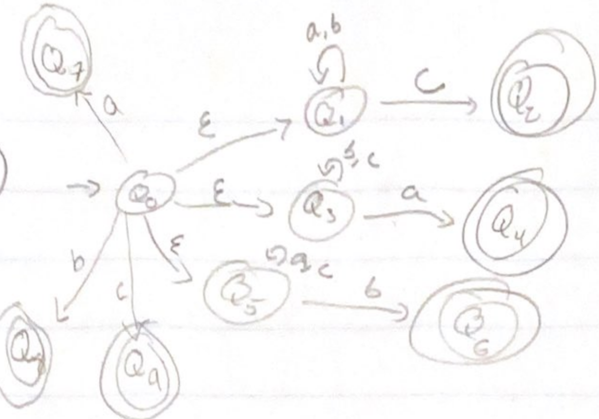
d)	Are members	Are not members
	1. aaa	1. a
	2. aaaaaa	2. aaaa

Explanation: This expression accepts any # of set of 3 a's, and nothing less than this.

e)	Are members	Are not members
	1. abbaabbbaaa	1. aaaaaa
	2. aabbaa	2. abb

Explanation: This expression accepts any combination of a & b prior to the set in stone characters in the expression (a & b).

5)



Explanation: This NFA allows for strings {a, b, c} where it accepts only characters of length 1 of a, b, or c or strings {a, b, c} that end in a character that had not yet occured in the string. w/ length n This can also be generalized for all languages using this same concept b/c all you have to do is change the characters for whichever chars are in the language and change the # of states that corresponds to the # of combinations to get a nonrepeating last char.