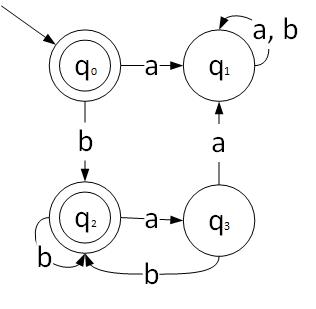
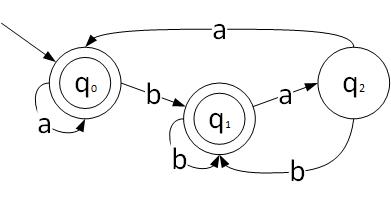
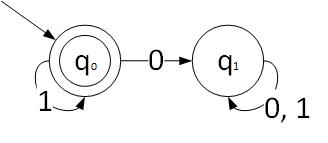
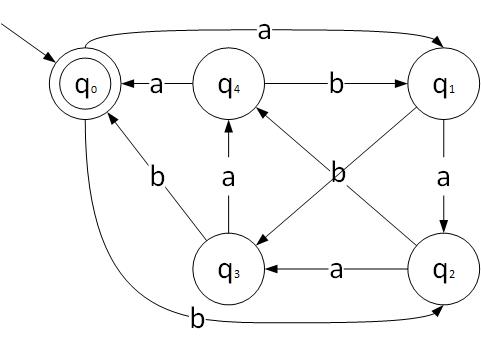
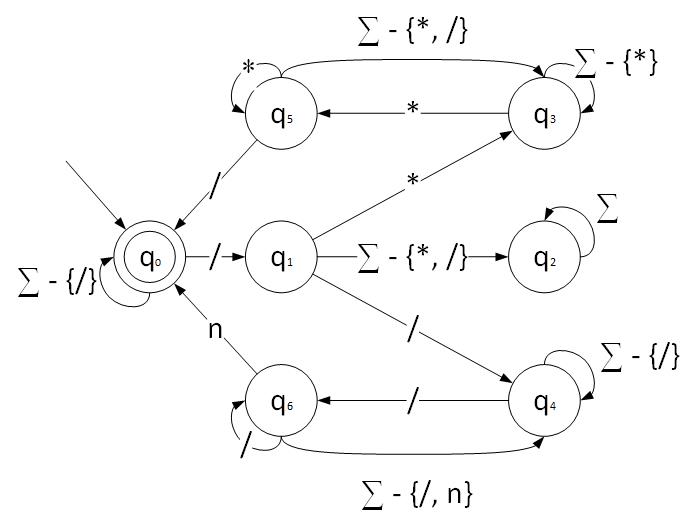
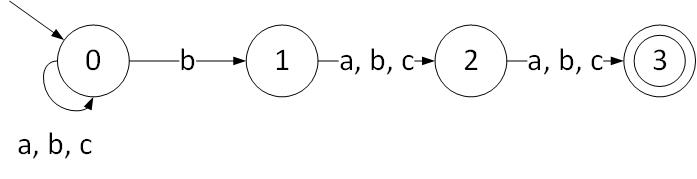
3331 Assignment 1

Zaid Albirawi  
250626065

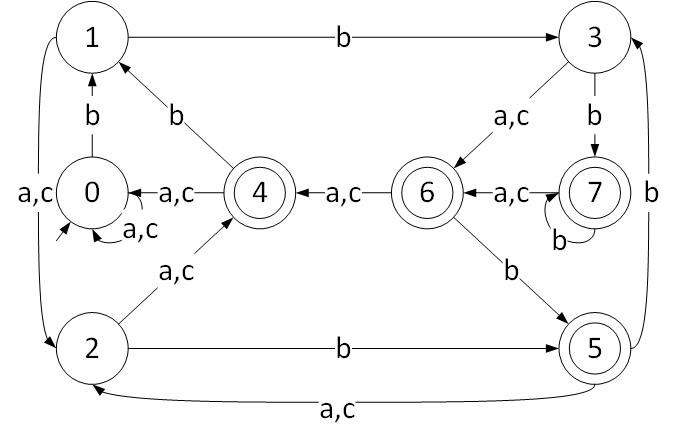
1. Build DFSMs for the following languages. Explain why your construction is correct.  
   1.   
      The construction is correct because, if *w* doesn’t start with a ‘*b*’ then a dead state is reached because that implies that the expression started with an ‘*a*’ and that ‘*a*’ is not preceded by a ‘b’. Else if there exists an ‘*a*’ that is followed by another, then go to dead state because the ‘*a*’ is not followed by a ‘*b*’. Else accept.
   2.   
      The construction is correct because the machine is always in acceptance state as long as the string does not terminate at q2, which implies that the last two characters were ‘ba’.
   3. 

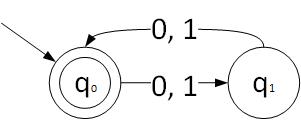
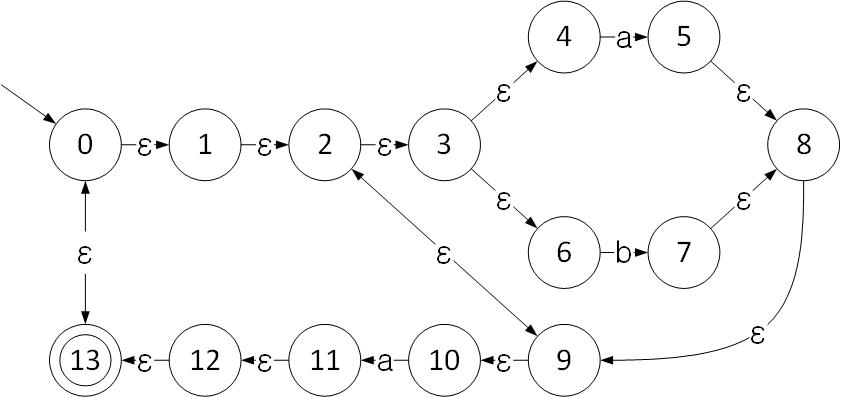
The construction is correct because if there exists a ‘0’ in the expression, then there exists a prefix of the expression, *w*, that will end with ‘0’.

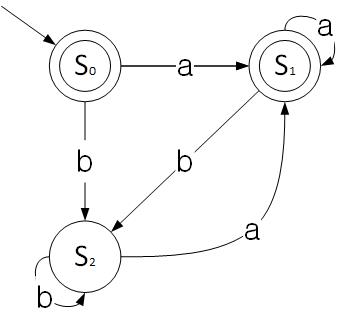
* 1.   
     The construction is correct because the machine is only in acceptance state when the sum of. Which is when the string is empty or when the sum is a multiple of 5. Note: the numbers of nodes reflect the value for
  2. C++ comments: /\* … comment … \*/ or // … comment … \n.  
       
       
     The construction is correct because the machine is only in acceptance state when there are no comments, or when the comments are closed.

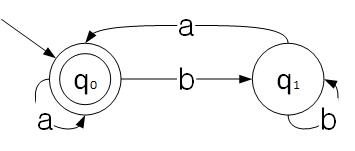
1. Consider the language:  
   1. Build a NDFSM for *L*.  
        
      
   2. Transform it into a DFSM
      * eps (0) = {0}
      * eps (1) = {1}
      * eps (2) = {2}
      * eps (3) = {3}  
        Since ‘a’ and ‘c’ are in the same equivalence class, then let us refer to them as one character, ‘e’.
      * ({0}, ‘e’) = {0}

Let:

* {0} = 0
* {0, 1} = 1
* {0, 2} = 2
* {0, 1, 2} = 3
* {0, 3} = 4
* {0, 1, 3} = 5
* {0, 2, 3} = 6
* {0, 1, 2, 3} = 7
  + - ({0}, ‘b’) = {0, 1}
    - ({0, 1}, ‘e’) = {0, 2}
    - ({0, 1}, ‘b’) = {0, 1, 2}
    - ({0, 2}, ‘e’) = {0, 3}
    - ({0, 2}, ‘b’) = {0, 1, 3}
    - ({0, 1, 2}, ‘e’) = {0, 2, 3}
    - ({0, 1, 2}, ‘b’) = {0, 1, 2, 3}
    - ({0, 3}, ‘e’) = {0}
    - ({0, 3}, ‘b’) = {0, 1}
    - ({0, 1, 3}, ‘e’) = {0, 2}
    - ({0, 1, 3}, ‘b’) = {0, 1, 2}
    - ({0, 2, 3}, ‘e’) = {0, 3}
    - ({0, 2, 3}, ‘b’) = {0, 1, 3}
    - ({0, 1, 2, 3}, ‘e’) = {0, 2, 3}
    - ({0, 1, 2, 3}, ‘b’) = {0, 1, 2, 3}
  1. Build an equivalent regular expression from one of the two FSM above. (*Hint:* it makes a difference which FSM you choose.)  
       
     Using the NDFSM:   
       
     Regular expression:

1. For the following languages, *L,* describe the equivalence classes of *L.* If there are finitely many classes, then build a minimal DFSM that accepts *L*.
   1. [0] [a, b, aaa, bbb] (odd length strings)   
      [1] [aa,]   
      [2] [bb,]   
      [3] [aaaa,]   
      [4] [bbbb,]   
      [5] [abba,]   
        
      Therefore, each even length string will form its own equivalence class because the reversal of each word is only specific to that word (There does not exists a 2 words that have the same reversal word, unless they are identical). The language is irregular.
   2. If an odd number is defined by 2n + 1, and an even number is defined by 2n. Then if we are looking for both theto be either even or odd, then the length of the string will have to be even because, odd + odd = 2n + 1 + 2n + 1 = 2(2n) + 2 = 2k, where k = n + 1 and even + even = 2n + 2n = 4n = 2i, where i = 2n. Therefore,   
        
      [0] [00, 11, 000111, 110010, …], (even length, )  
      [1] [0, 1, 001, 110, 11001, 00110…], (odd length)  
        
      
2. Consider the regular expression.  
   1. Construct an NDFSM that accepts. (You can use Thompson’s construction but you don’t have to.)  
      
   2. Transform it into DFSM.
      * eps (0) = {0, 1, 2, 3, 4, 6, 9, 10, 13}
      * eps (1) = {1, 2, 3, 4, 6, 9, 10}
      * eps (2) = {2, 3, 4, 6, 9, 10}
      * eps (3) = {3, 4, 6}
      * eps (4) = {4}
      * eps (5) = {2, 3, 4, 5, 6, 8, 9, 10}
      * eps (6) = {6}
      * eps (7) = {2, 3, 4, 6, 7, 8, 9, 10}
      * eps (8) = {2, 3, 4, 6, 8, 9, 10}
      * eps (9) = {2, 3, 4, 6, 9, 10}
      * eps (10) = {10}
      * eps (11) = {0, 1, 2, 3, 4, 6, 9, 10, 11, 12, 13}
      * eps (12) = {0, 1, 2, 3, 4, 6, 9, 10, 12, 13}
      * eps (13) = {0, 1, 2, 3, 4, 6, 9, 10, 13}

Let S0 = eps(0) = {0, 1, 2, 3, 4, 6, 9, 10, 13}, S1 = eps(5, 11) = {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13}, and S2 = eps (7) = {2, 3, 4, 6, 7, 8, 9, 10}.  


* 1. Minimize it.  
       
        
        
     Let q0 = {0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13}, q1 = 2, 3, 4, 6, 7, 8, 9, 10}  
     

1. For each of the following languages *L*, prove whether *L* is regular or not.  
   1. .  
        
      If, then,

Hence, .  
Using the pumping theorem, let k = i. Such that .  
Let , where, and , where .  
Let , therefore,, since , then . , contradiction. The language is not regular.

* 1. .  
       
     Let, where , and and let   
     Therefore, .  
       
     Using the pumping theorem,   
     let , where , where .  
     Hence, .  
     Furthermore, let , therefore,   
     Finally, . Contradiction, the number of a’s is imbalanced on each side, the language is not regular.

1. Show that the following problem is decidable: Given and a regular expression, does the language defined by contain all the even length strings in?  
     
   Firstly, let be the expression that accepts all even strings in in.  
   Secondly, construct two FSMs and such that, and in.  
   If the, then return true, otherwise, return false. A representation of ,  
     
   