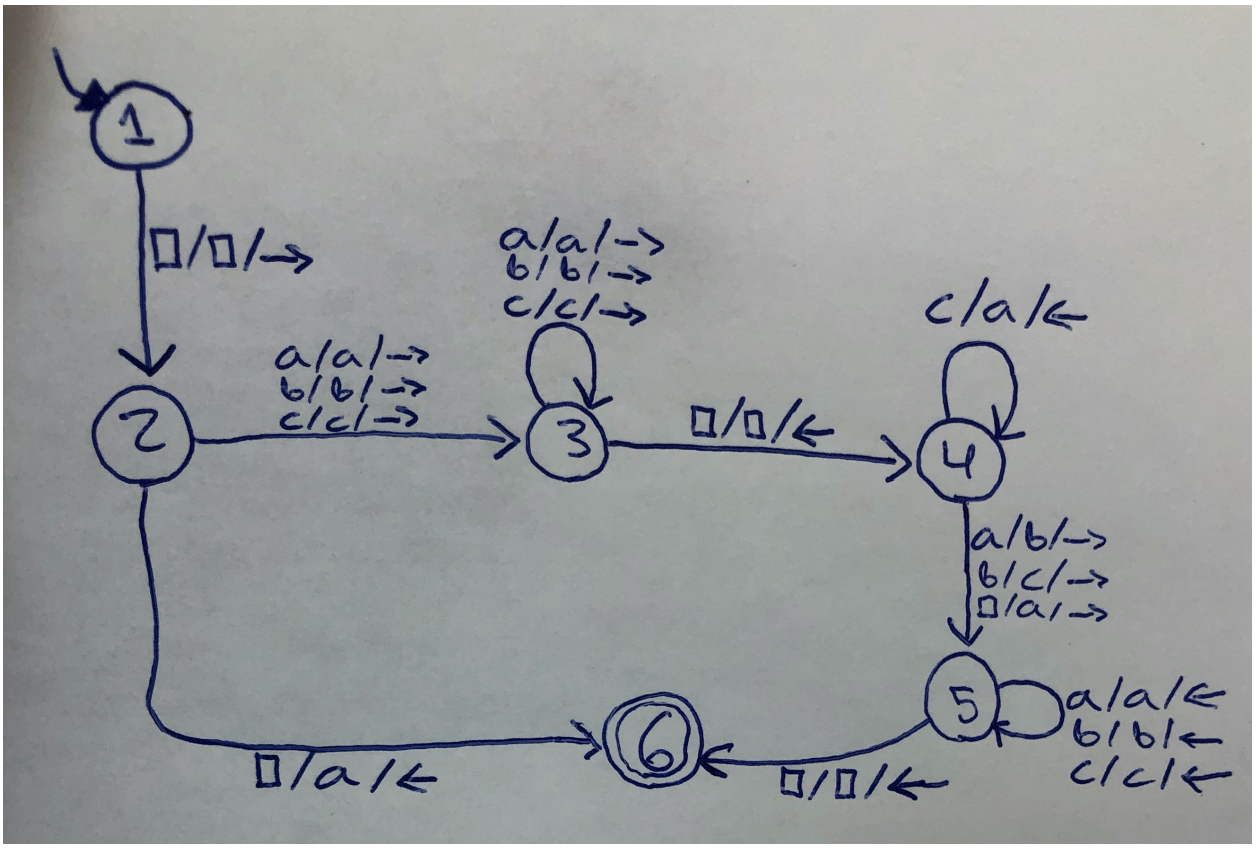
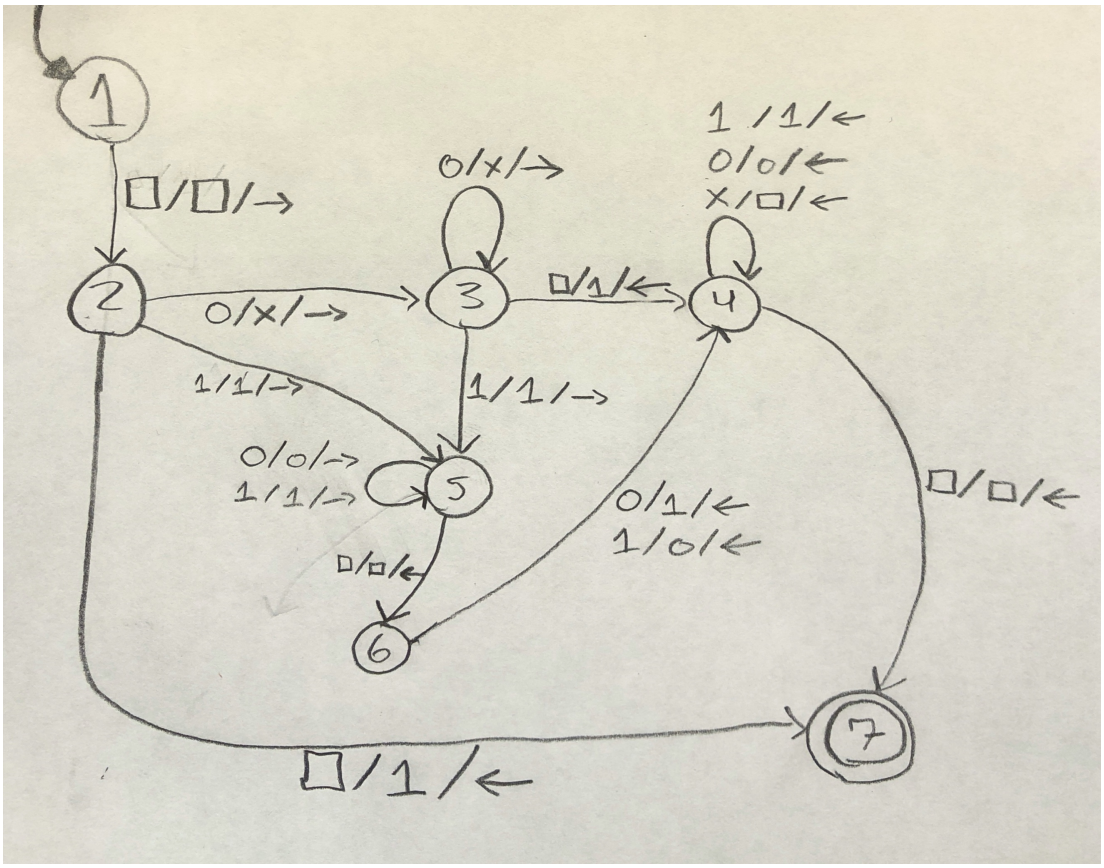
Name: Firas Aboushamalah
Course: CS3331
Student#: 250920750

Assignment 3

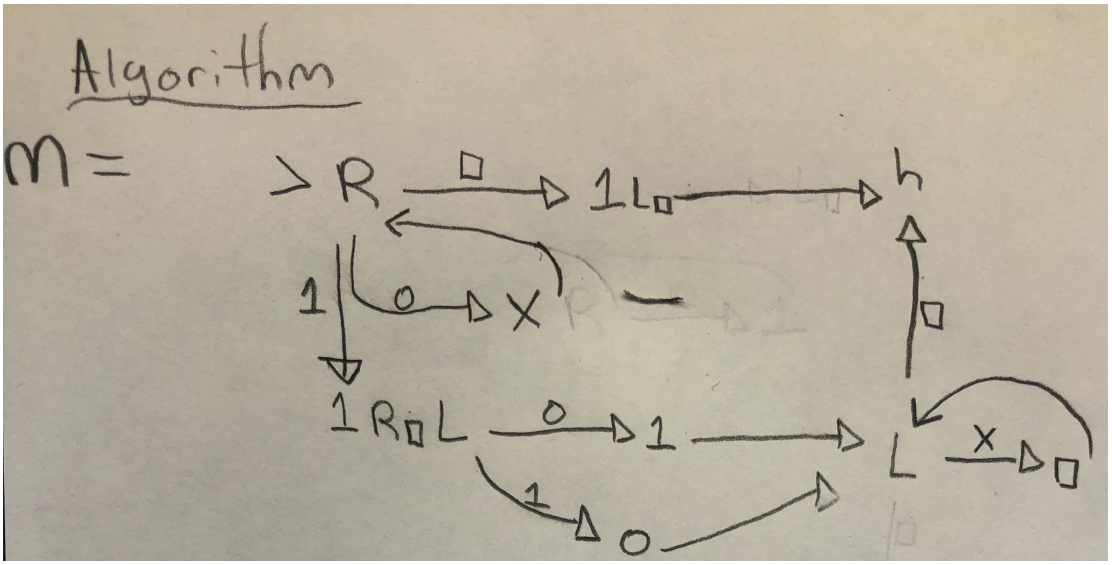
1. (10pt) Consider the alphabet $\Sigma = \{a, b, c\}$ and define the function succ : $\Sigma* \rightarrow \Sigma*$, succ(w) is the word immediately following w in lexicographic order. Construct a deterministic Turing machine M that computes the function succ, that is, M starts with the initial configuration (s, □w) and halts with the configuration (h,□succ(w)). Describe M in details using a directed graph whose edges are labelled by transitions (such as the one in Example 17.2, p. 268 of textbook).



2. (10pt) Construct a deterministic Turing machine M that adds one to its binary input if it is even and subtracts one if it is odd. M starts with the initial configuration (s,□w), where w ∈ {0,1}∗; the binary input w is interpreted as an integer number. Possible leading 0's have to be removed as well. The machine halts in the appropriate configuration (h, □(w ± 1)(2)), where w(2) is the binary representation of w.

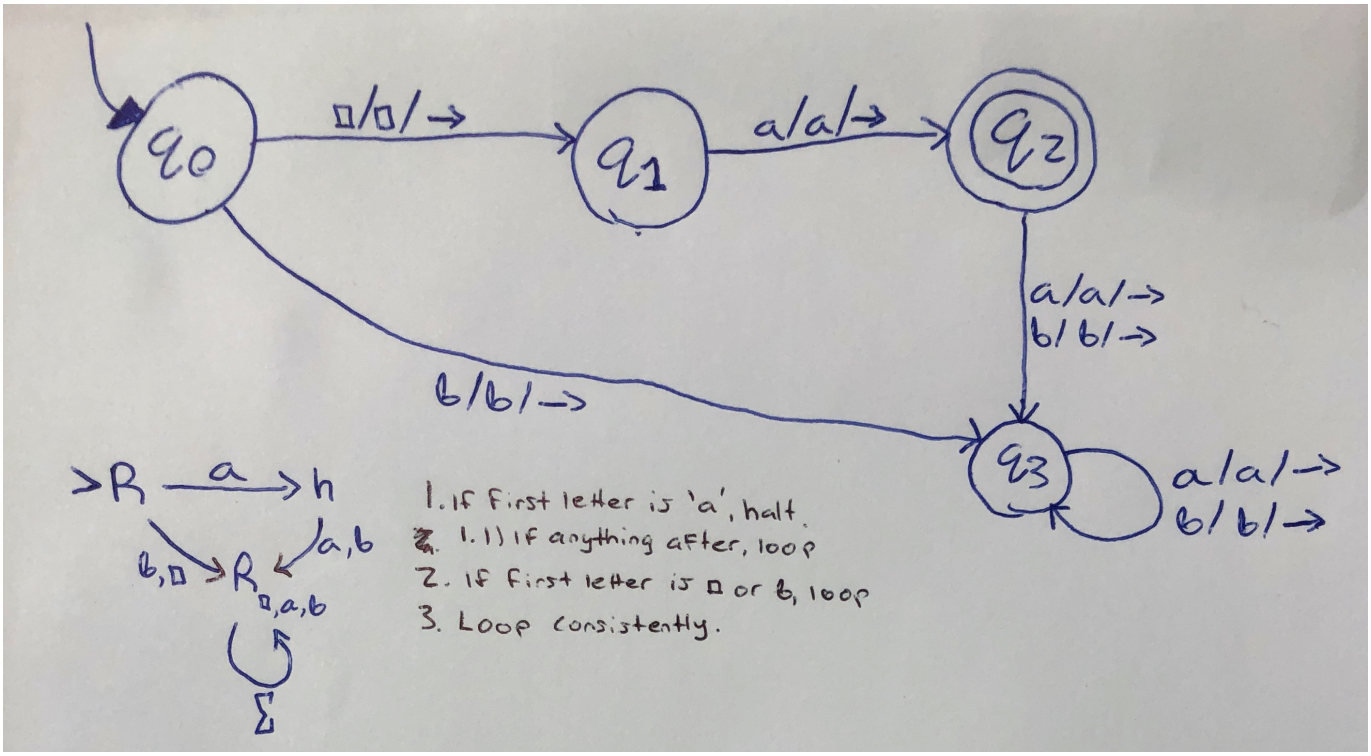


1. Check if next symbol after head is □
   1.1. Write 1 then return to beginning of string and HALT
2. LOOP
   2.1. If next digit is 0, replace with X
   2.2. If next symbol is □, write a 1
   2.3. Remove all "X" then go to beginning of string and HALT
3. Encountered a 1; scan until you reach the last number in the string
   3.1. If 0, replace with 1
      3.1.1. remove all "X" then go to beginning of string and HALT
   3.2. if 1, replace with 0
      3.2.1. remove all "X" then go to beginning of string and HALT



3. (20pt) Construct a Turing Machine M that semidecides, but does not decide, each of the following languages over the alphabet Σ = {a, b}: In each case, describe M using the macro language.

   (a) L1 = {a}

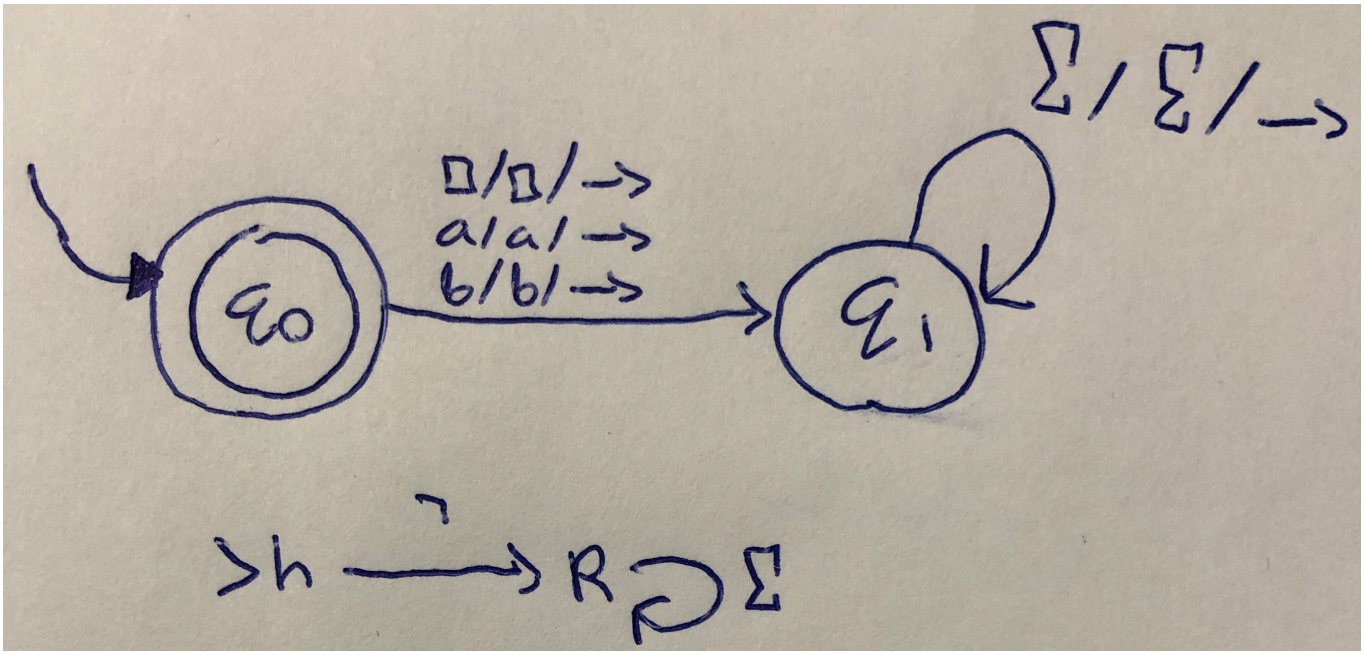


(b) L2 = Σ^*

This cannot be done because the set of all strings in the alphabet {a, b} means that the length of the input is infinite. The Turing Machine cannot account for this because it would have to accept an infinite amount of strings. Therefore, there is no string that is NOT accepted meaning this is an undecidable problem and therefore does not have a Turing Machine associated with it.

Turing Machines are simply Finite State Machines with a tape (must have a finite length input).

(c) L3 = ∅.



4. (20pt) Describe in clear English a Turing machine that semidecides the language
   L = {< M >| M accepts the binary encodings of at least 3 prime numbers}

Semidecidability: Given a Turing Machine and an set of input(s) derived from a language, the Turing Machine will halt if the string is in the language. Otherwise, it will reject or loop forever.

In binary, the encoding of a prime number will always begin and end with one; except for 2 which is encoded as 10. This will be the special case. Therefore, our turing machine will examine the first number in the string, if it is a one, it moves on to the next state. It will

traverse until the end of the string until it reaches the last number, then it will check if this number is a one. If it is, it will halt and accept. If it does not, it will enter a loop as the number is **not** prime. Now, for the special case of 2, the Turing Machine will check if the number that starts with 1 strictly ends with 0.

Therefore we define a semideciding TM M'(<M>) as follows:

1. Run M on the input.
2. If the input in M is the binary encoding of a prime number, halt and accept.
3. If it does not, loop
4. In the loop, keep checking if there are more prime numbers
    a. If they are equal to 3 encodings then halt and accept

5. (20pt) Is the set SD closed under:

    (a) Intersection?

    If we have L1 and L2 and both are SD, then L1 $\cap$ L2 is closed as it is also SD. For example, if we have L1 = SD and L2 = SD, and Turing Machines M1 and M2 for L1 and L2 respectively, then we can simulate the same input x $\in$ L1 $\cap$ L2 by running it through each machine SIMULTANEOUSLY like so:

    1. Copy input to tape 1
    2. Copy input to tape 2
    3. Run both M1 and M2 at the same time, one step after another
    4. If either machine accepts, copy the input and put it on the opposite tape and run that machine and see if it accepts at the same state
        a. If it accepts, then both machines accepted then x is in L1 $\cap$ L2
    5. If either machine rejects or loops, then check the other machine to see if it accepts or loops. However, they must both act the same.

    (b) Concatenation?

    1. Similarly, we use the same language setup (L1 and L2 & M1 and M2) from question 5a. In order to find out if some input is in the concatenation of L1 and L2, we must first assume where we can divide the input x into two parts: x1x2. Now, we run M1 on x1 and M2 on x2. If x is in the language of L1L2, M1 and M2 will both eventually accept. Otherwise, if x is not in L1L2, then there is no place which we can divide x = x1x2 which causes a rejection or infinite loop in the machines.

6. 20pt) Let L1 and L2 be two languages that are not decidable.

a) **Is it possible that L1 − L2 is regular and L1 − L2 ≠ ∅? Prove your answer**.

Yes, this is true. We can show by example:

If we take the undecidable languages:

$\mathbf{L_1} = L_2 \cup \{a\}$
$\mathbf{L_2} = b^*$
$\mathbf{L_1 - L2} = (b^* \cup \{a\}) \cap \neg b^*$ // *(X-Y=X ∩ ¬Y)*
$= \neg b^* \cap b^* \cup \neg b^* \cap a$
$= \neg b^* \cap a$
$= a$

Therefore we know that the set {a} is a regular language and it is not equal to the empty set.

b) Is it possible that L1 ∪ L2 is decidable but L1 ≠ ¬L2? Prove your answer.