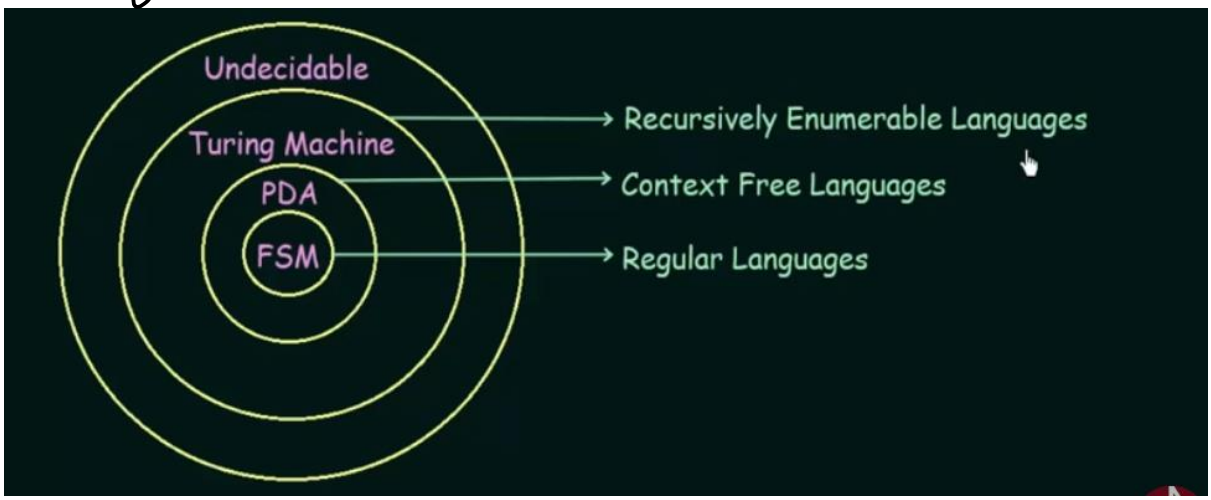
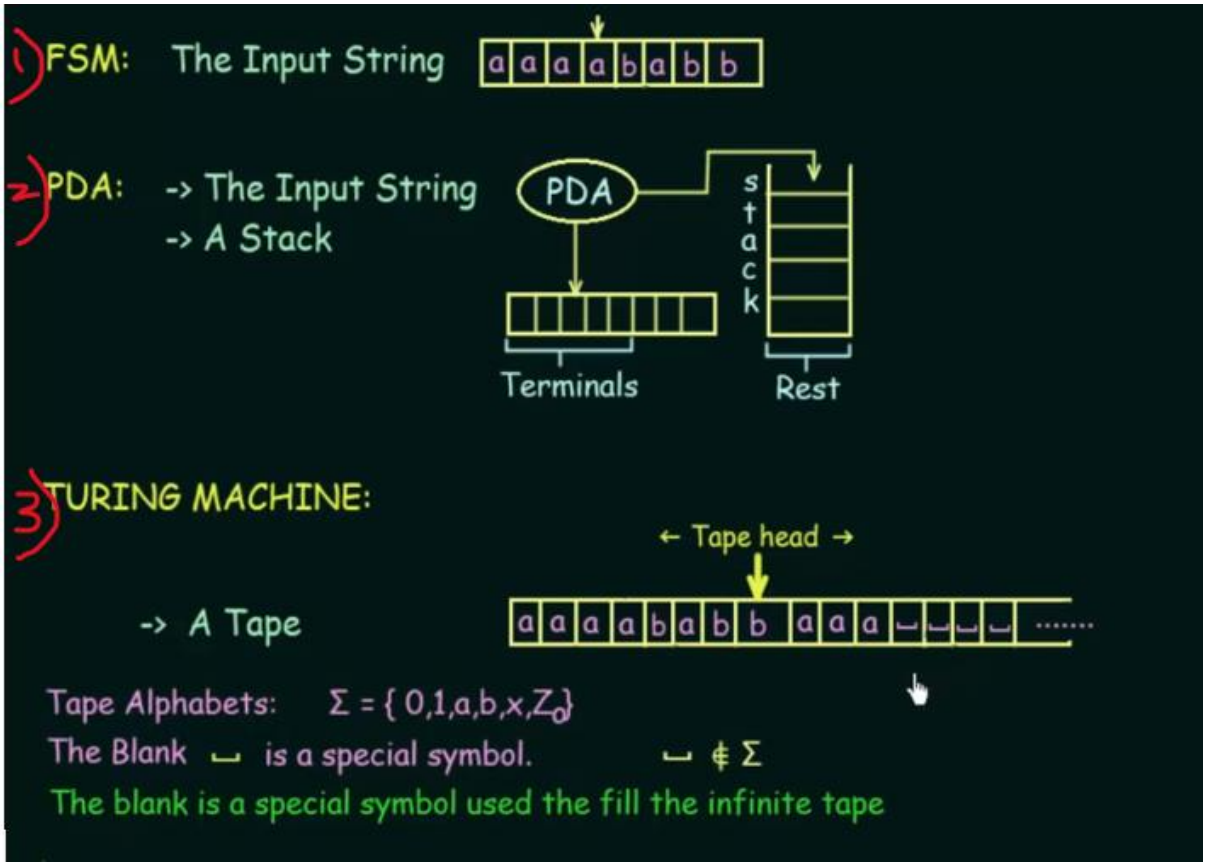
Accepting

Computing

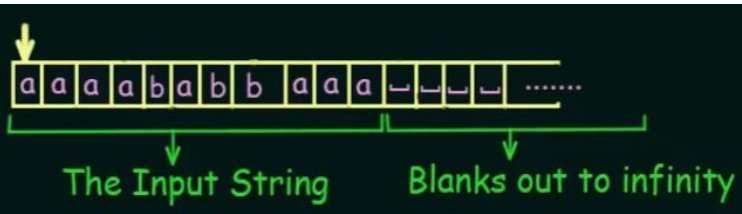
Turing Machine Intro and Basics

Turing Machine Accepting Device



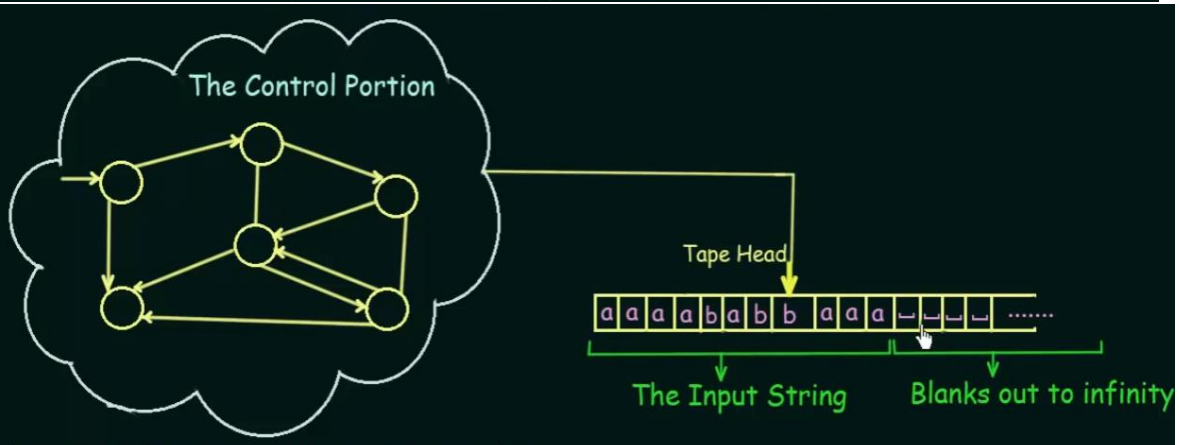


Initial Configuration:



Operations on the Tape:

-> Read /Scan symbol below the Tape Head
-> Update / Write a symbol below the Tape Head
-> Move the Tape Head one step LEFT
-> Move the Tape Head one step RIGHT

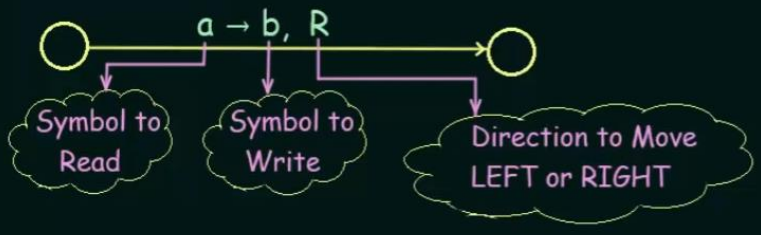


The Control Portion similar to FSM or PDA
The PROGRAM
It is deterministic

Rules of Operation - 1

At each step of the computation:
-> Read the currect symbol
-> Update (i.e. write) the same cell
-> Move exactly one cell either LEFT or RIGHT

If we are at the left end of the tape, and trying to move LEFT, then do not move. Stay at the left end



If you dont want to update the cell,
JUST WRITE THE SAME SYMBOL

Rules of Operation - 2

-> Control is with a sort of FSM

-> Initial State

-> Final States: (there are two final states)

1) The ACCEPT STATE

2) The REJECT STATE

-> Computation can either

1) HALT and ACCEPT

2) HALT and REJECT

3) LOOP (the machine fails to HALT)

A Turing Machine can be defined as a set of 7 tuples

$$(Q, \Sigma, \Gamma, \delta, q_0, b, F)$$

$Q \rightarrow$ Non empty set of States

$\Sigma \rightarrow$ Non empty set of Symbols

$\Gamma \rightarrow$ Non empty set of Tape Symbols

$\delta \rightarrow$ Transition function defined as

$$Q \times \Sigma \rightarrow \Gamma \times (R/L) \times Q$$

$q_0 \rightarrow$ Initial State

$b \rightarrow$ Blank Symbol

$F \rightarrow$ Set of Final states (Accept state & Reject State)

Thus, the Production rule of Turing Machine will be written as

$$\delta (q_0, a) \rightarrow (q_1, y, R)$$

Read the input symbol **a** and write into the tape as **Y** and move to the **right.**

Turing's Thesis:

Turing's Thesis states that any computation that can be carried out by mechanical means can be performed by some Turing Machine.

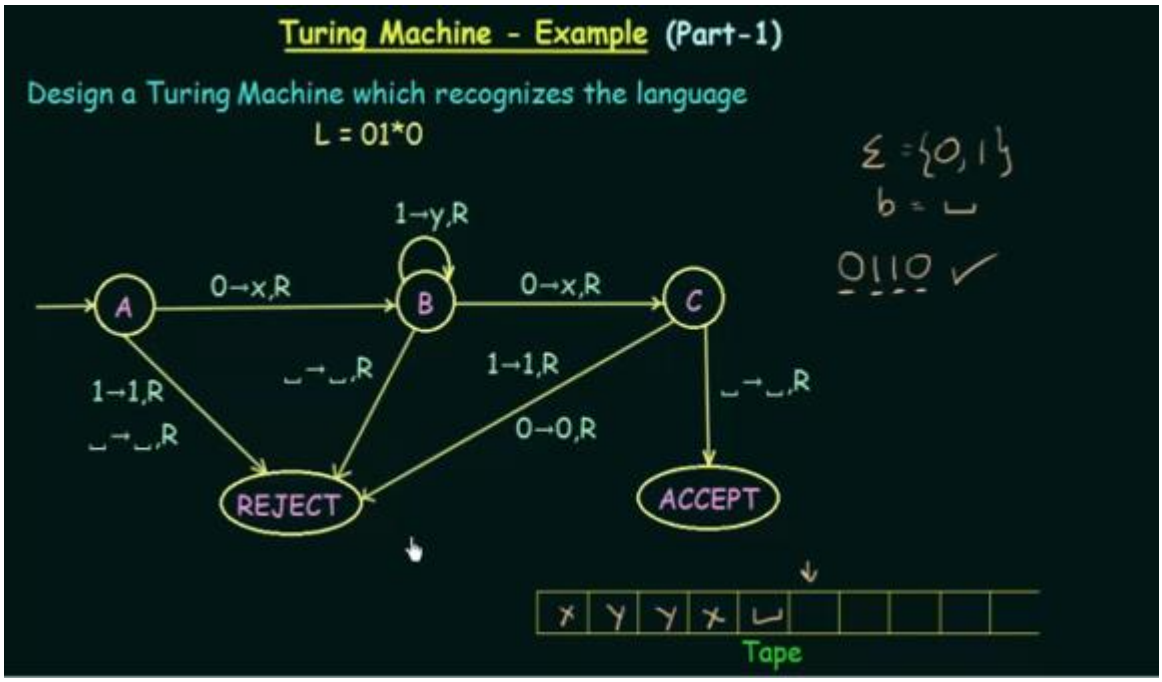
Few arguments for accepting this thesis are:

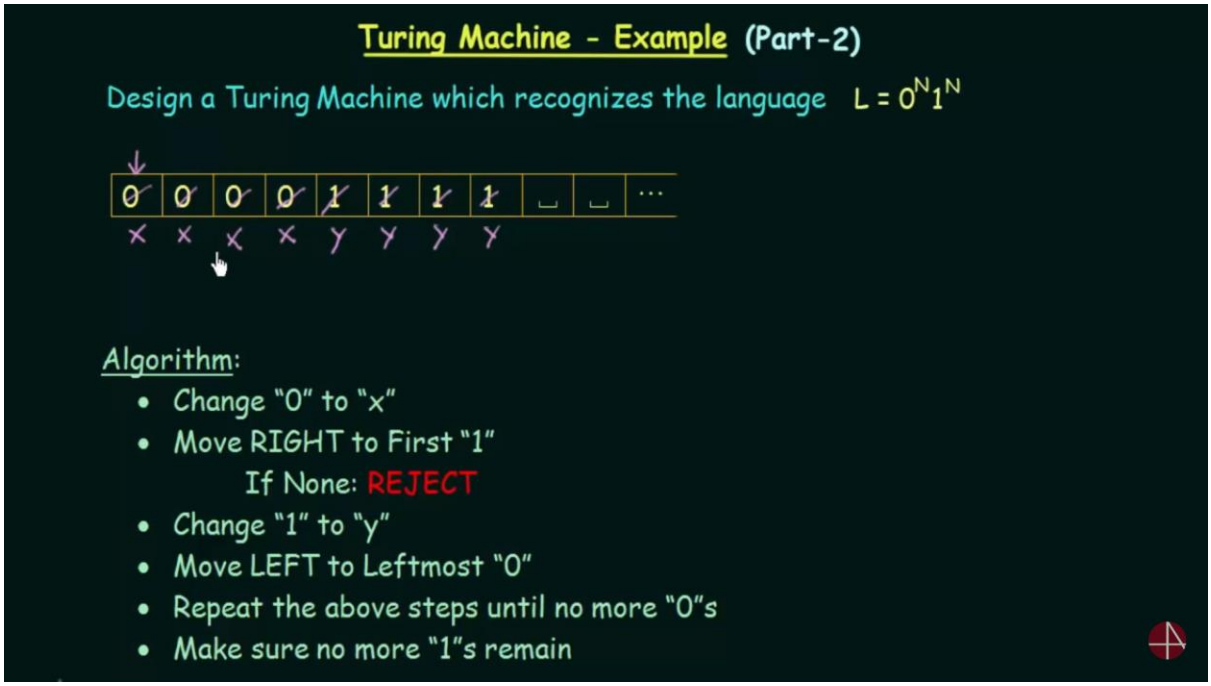
i. Anything that can be done on existing digital computer can also be done by Turing Machine.
ii. No one has yet been able to suggest a problem solvable by what we consider on algorithm, for which a Turing Machine Program cannot be written.

Recursively Enumerable Language:

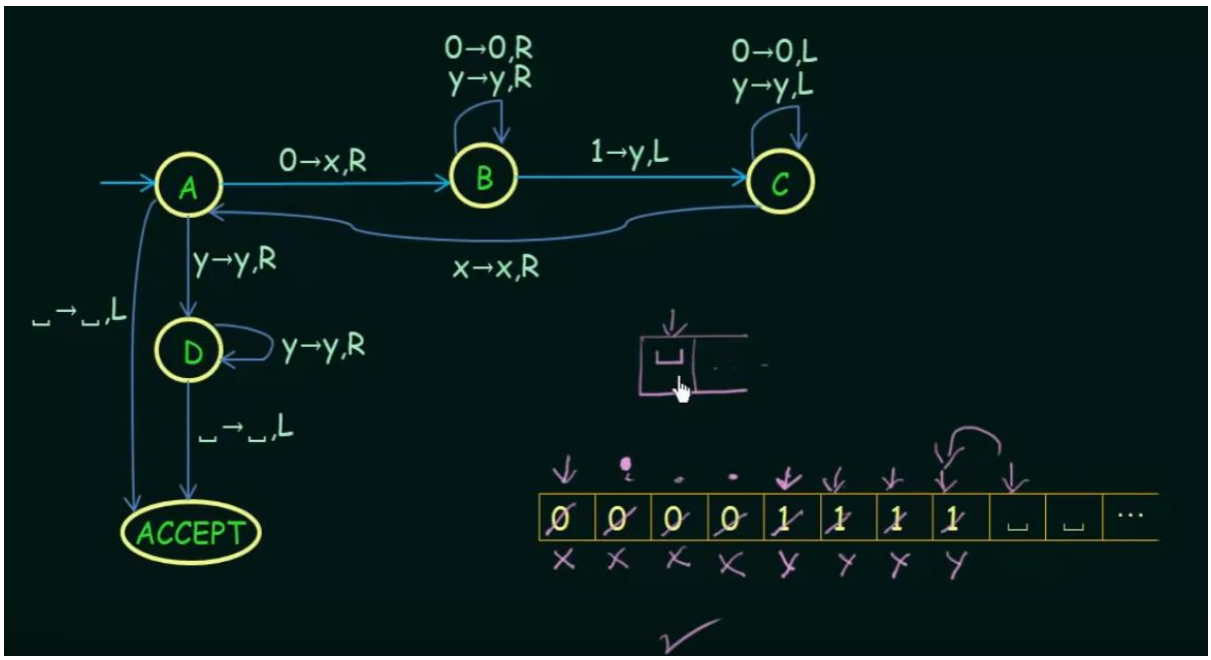
A Language L and ∑ is said to be Recursively Enumerable if there exists a Turing Machine that accepts it.

If I have a problem and I designed an algorithm to solve that particular problem, then definitely that Turing machine can also be designed to solve that problem.





found 1 extra 0 (i.e 5th 0, change to x and search for 1. If that 1 is not present then reject. There-fore **n(0)!=n(1)**



Hint:

After replacing all the strings(i.e 1 and 0) with x's and y's. Simply pass the states with x's and y's, if all the strings are exactly replaced by x and y, then your turing machine is correct for the given condition.

Design a Turing Machine that accepts the

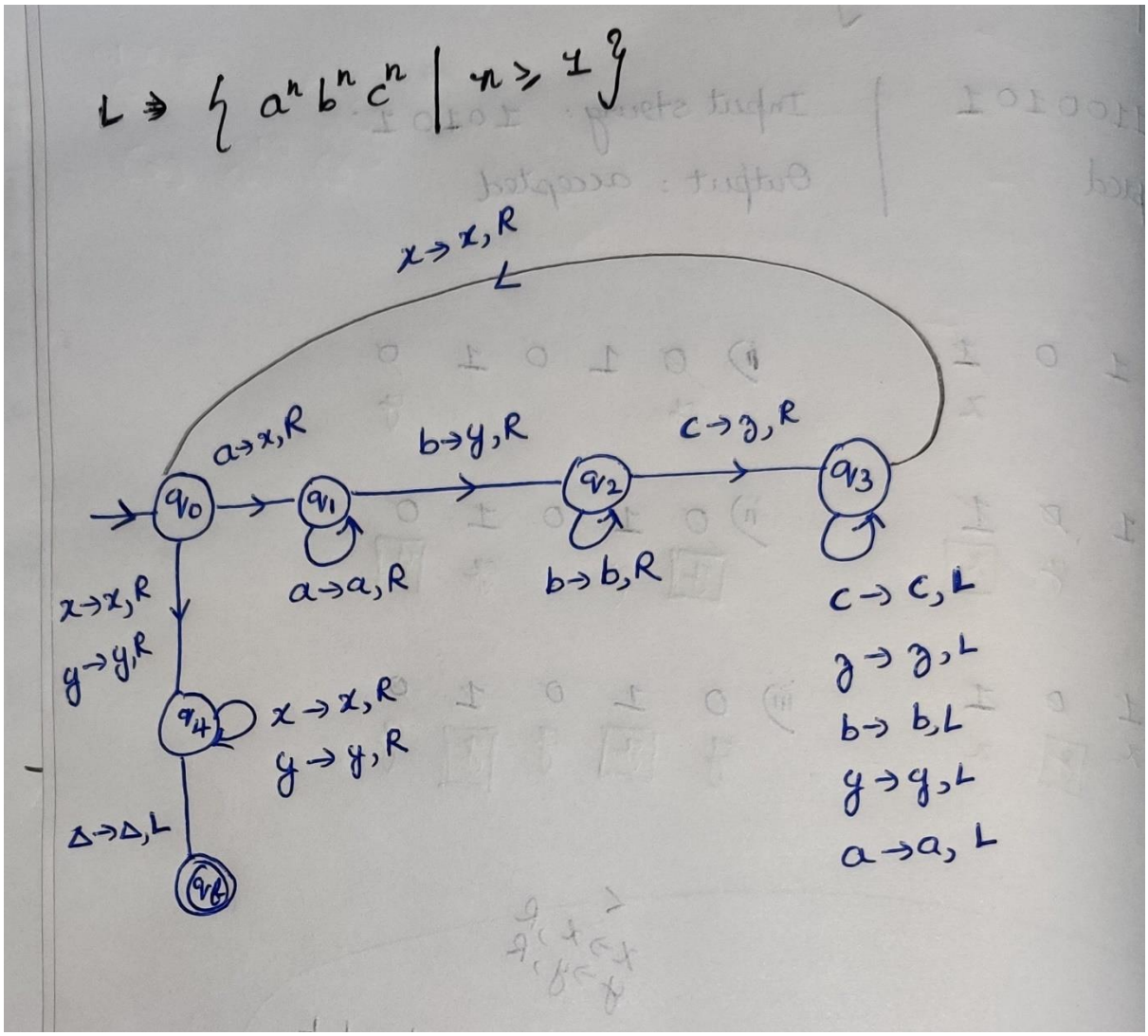
L={a*}

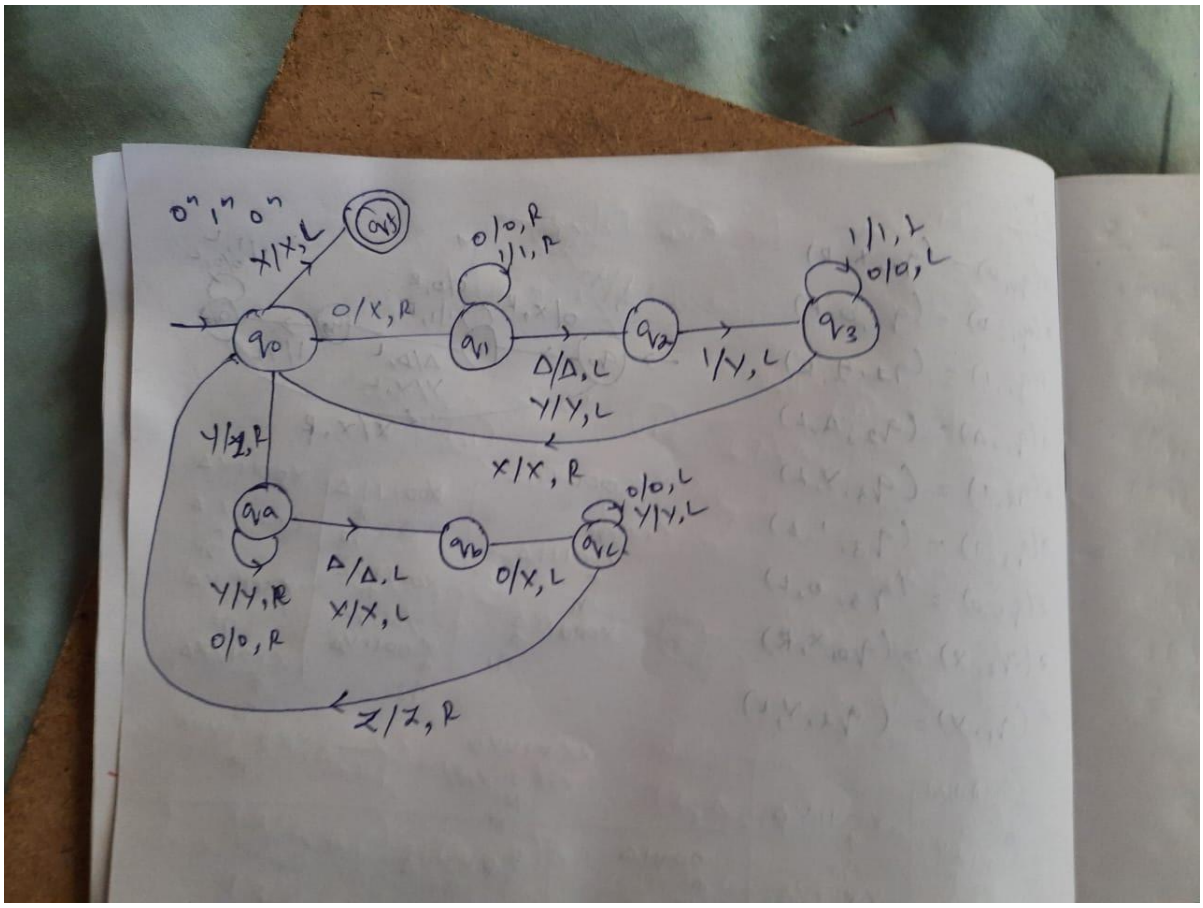
L = {a^+}

L = {a^n b^n | n>=1 }

Question

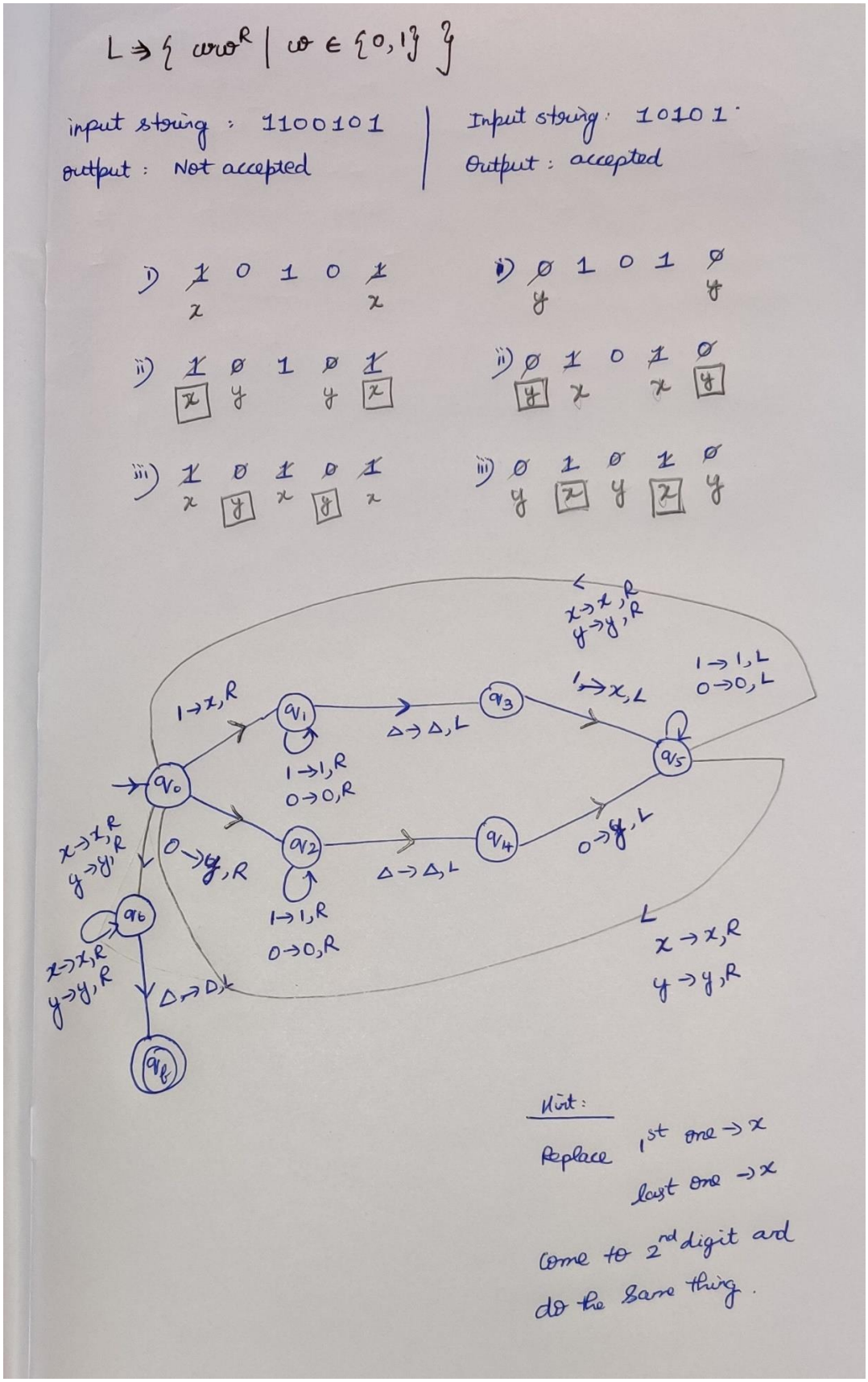
L = {a^n b^n c^n | n>=1 }

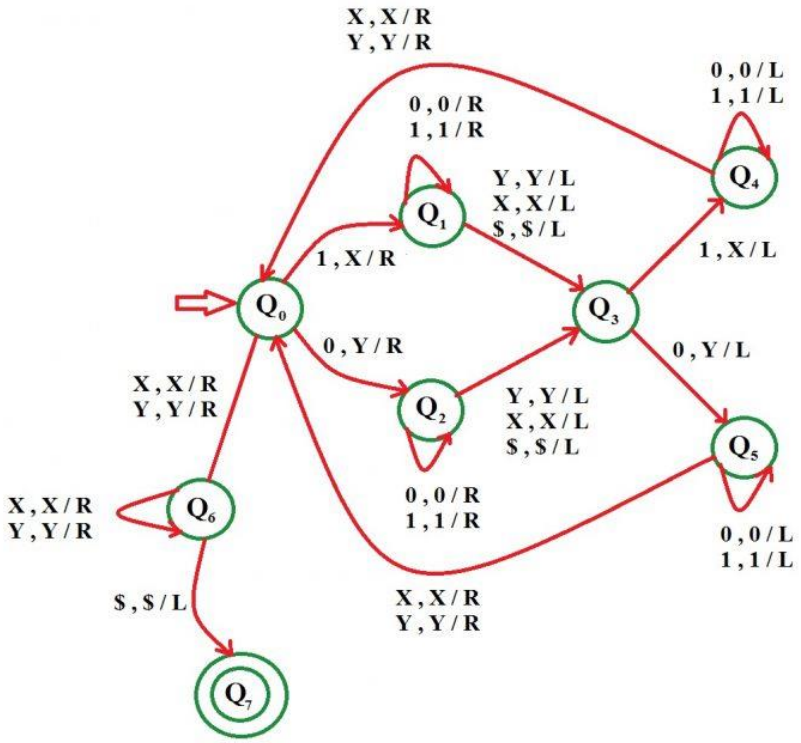



$0^n 1^n 0^n$
X/X, L
0/0, R
1/1, R
1/1, L
0/0, L
0/X, R
q_0
q_1
q_2
q_3
q_4
Δ/Δ, L
Y/Y, L
1/Y, L
Y/Y, R
X/X, R
q_a
q_b
q_c
0/0, L
Y/Y, L
Δ/Δ, L
X/X, L
0/X, L
Y/Y, R
0/0, R
Z/Z, R

Question

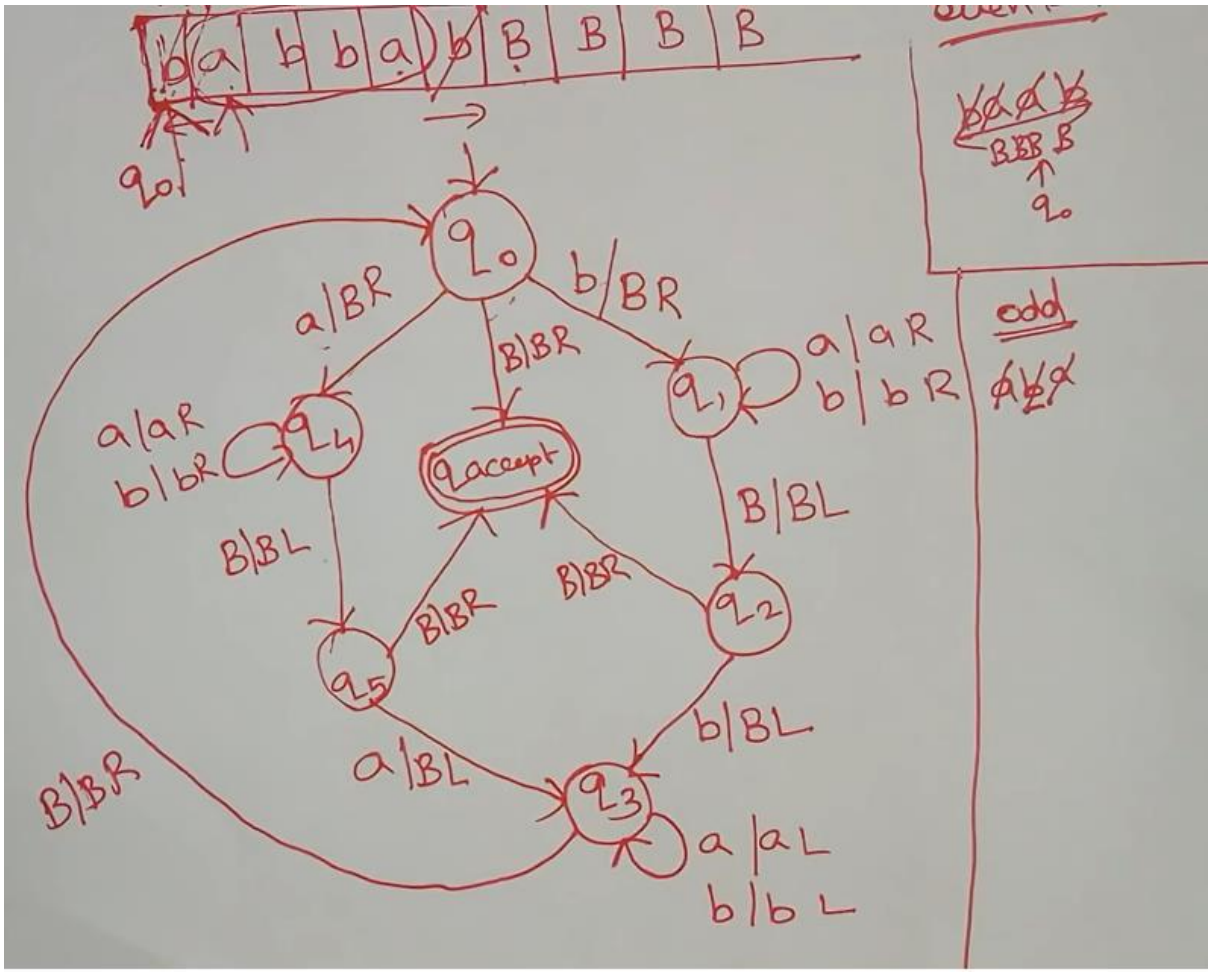
L = { ww^r | w belongs (0,1) }




X , X / R
Y , Y / R
0 , 0 / L
1 , 1 / L
0 , 0 / R
1 , 1 / R
Q4
Y , Y / L
X , X / L
$, $ / L
Q1
1 , X / L
1 , X / R
Q0
Q3
0 , Y / L
0 , Y / R
X , X / R
Y , Y / R
Q2
Y , Y / L
X , X / L
$, $ / L
Q5
X , X / R
Y , Y / R
Q6
0 , 0 / R
1 , 1 / R
0 , 0 / L
1 , 1 / L
X , X / R
Y , Y / R
$, $ / L
Q7

Question

Palindrome

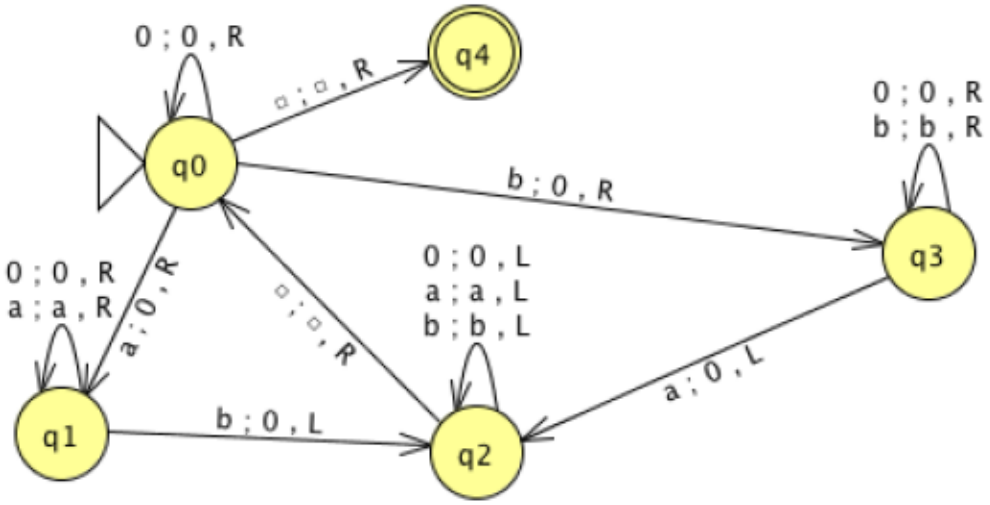


Even palindrome → abab

Odd palindrome → aba

Question

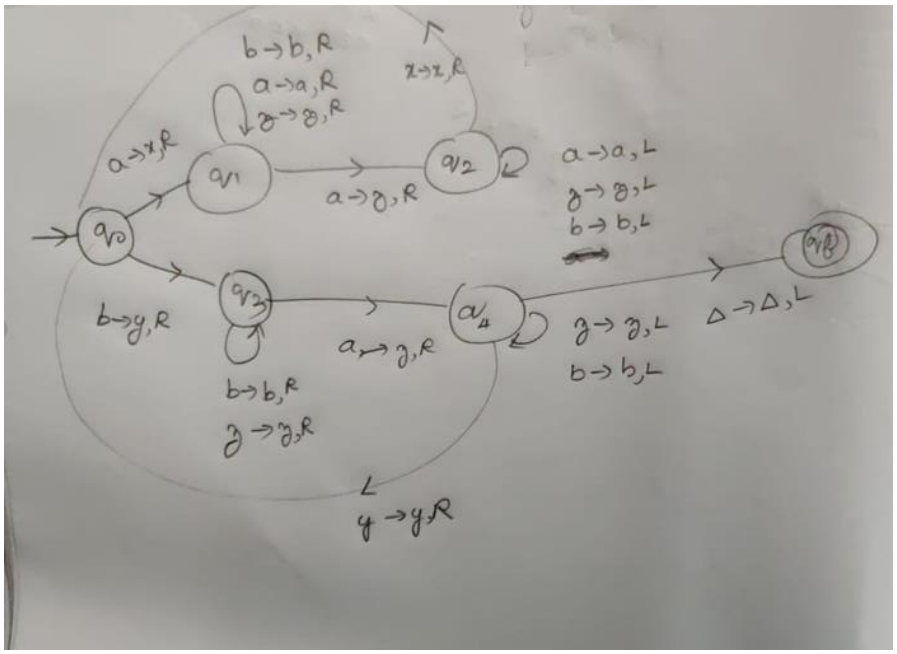
L = { n(a)=n(b) | w belongs (0,1) }



Question

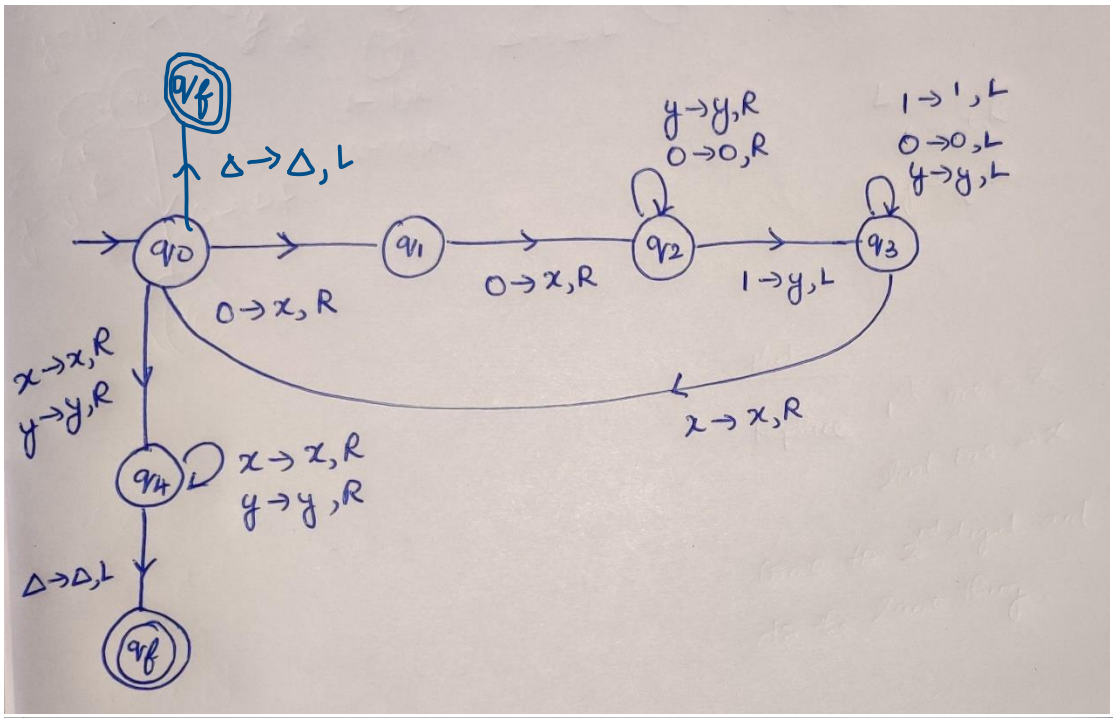
L = { a^n b^m a^(n+m)| n,m>=1 }

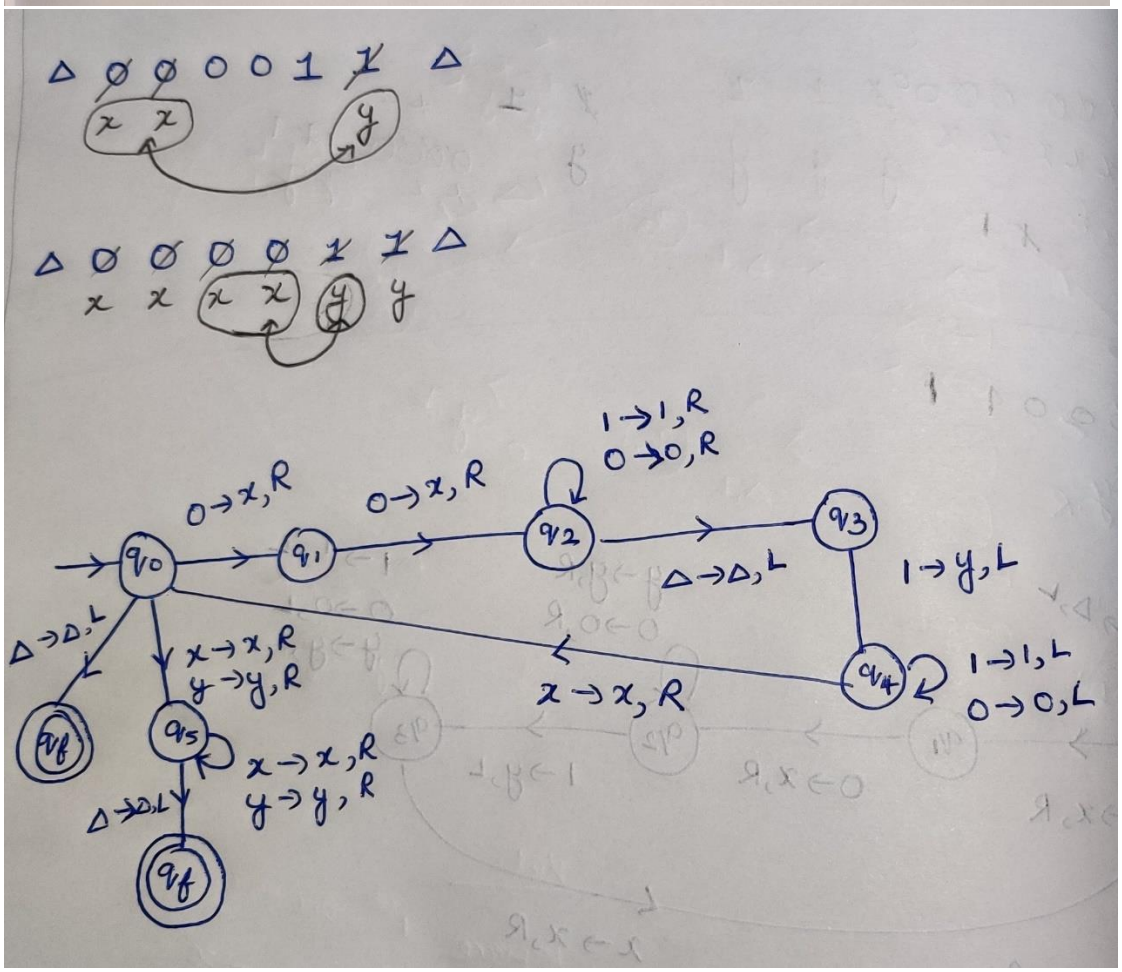
https://www.geeksforgeeks.org/construct-turing-machine-l-bm-anm-nm%e2%89%a51/?ref=rp



Question

L = { 0^2n 1^n | n>=0 }





Turing Machine as a computing device

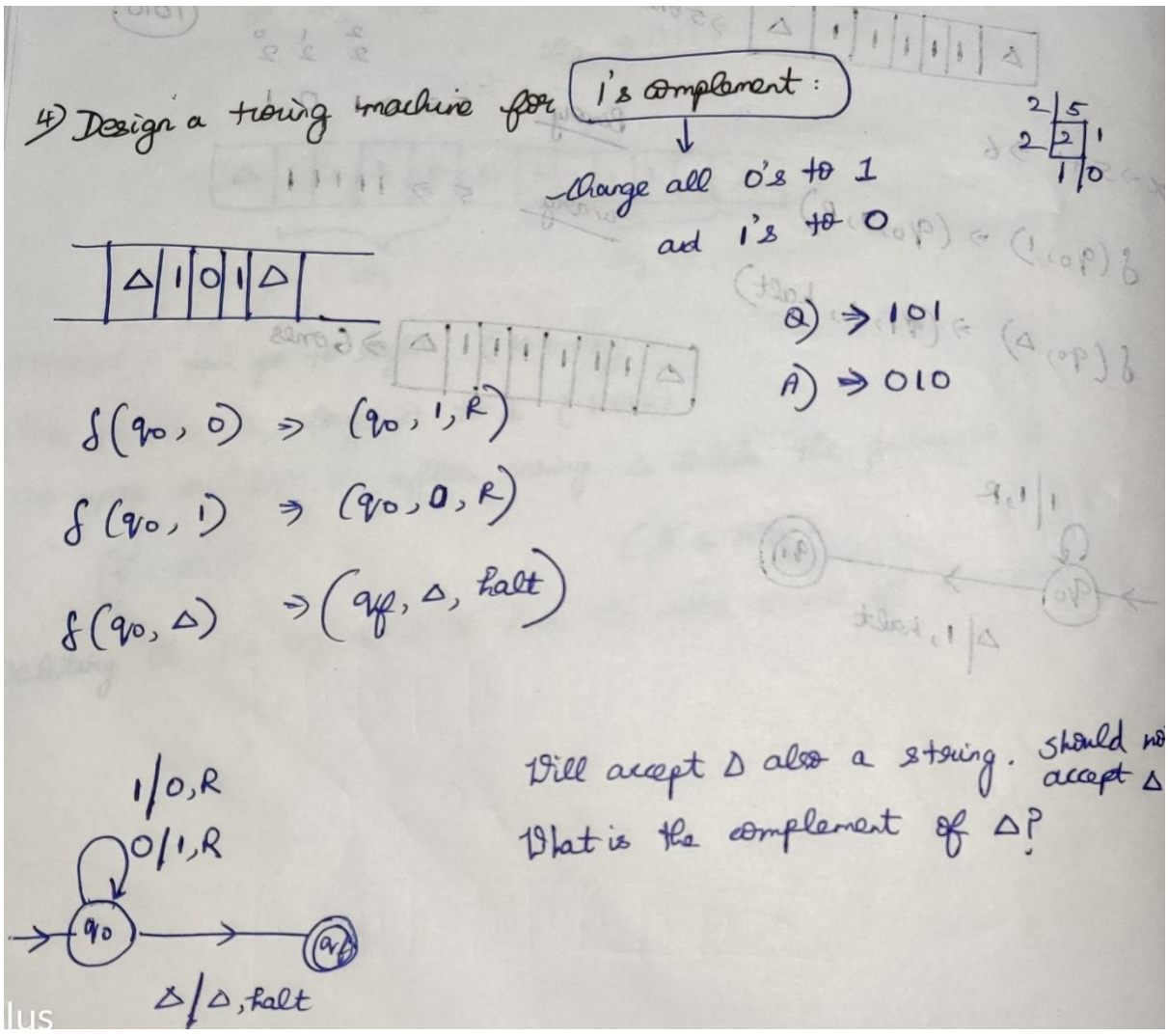
Design a turing machine for concatenating two strings
w1.w2=w
w1 → 11111
w2 → 1111

Design a turing machine for x+1

Design a turing machine for x+3

Question

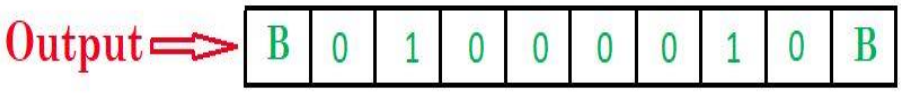
Design a Turing machine for 1's complement

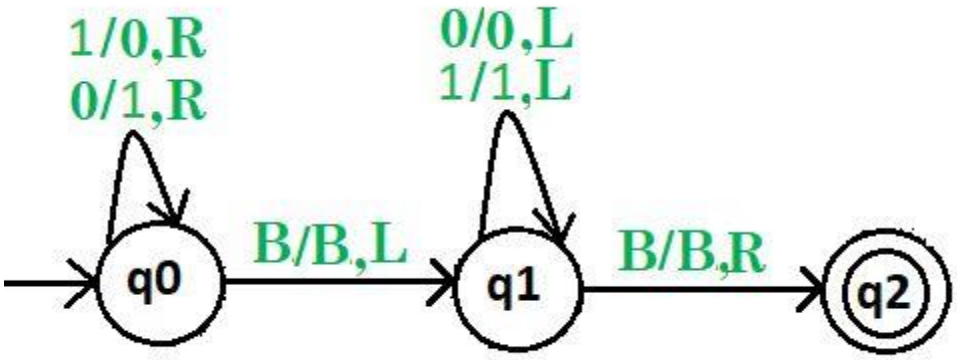


What is wrong ??
With no input, only accepting the blank input it will go to the final state.
There is no complement for the blank symbol.

Approach:

1. Scanning input string from left to right
2. Converting 1's into 0's
3. Converting 0's into 1's
4. Move the head to the start when BLANK is reached.





After traversing all the symbols(left to right) definitely traverse it once again(right to left) and keep the cursor at the first position of the string .

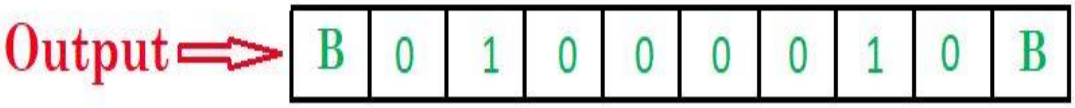
Question

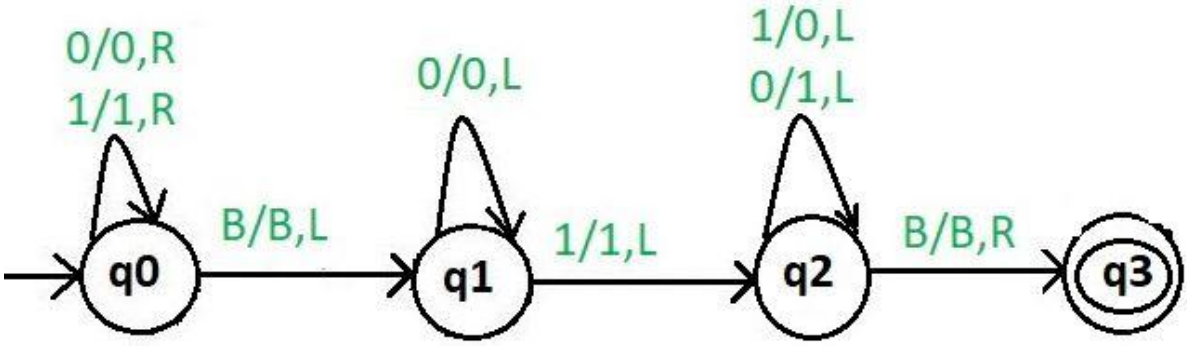
Design a Turing machine for 2's complement

(12) 1100 → 0100
(13) 1101 → 0011
(18) 10010 → 01110

Approach:

1. Scanning input string from right to left
2. Pass all consecutive '0's
3. For first '1' comes, do nothing
4. After that, Converting 1's into 0's and Converting 0's into 1's
5. Stop when BLANK is reached.





After traversing all the symbols(left to right) definitely traverse it once again(right to left) and keep the cursor at the first position of the string

Turing machine for addition(Unary representation of a number)

A number is represented in binary format in different finite automatas like 5 is represented as (101) but in case of addition using a turing machine unary format is followed. In unary format a number is represented by either all ones or all zeroes. For example, 5 will be represented by a sequence of five zeroes or five ones. 5 = 1 1 1 1 1 or 0 0 0 0 0. Lets use zeroes for representation.

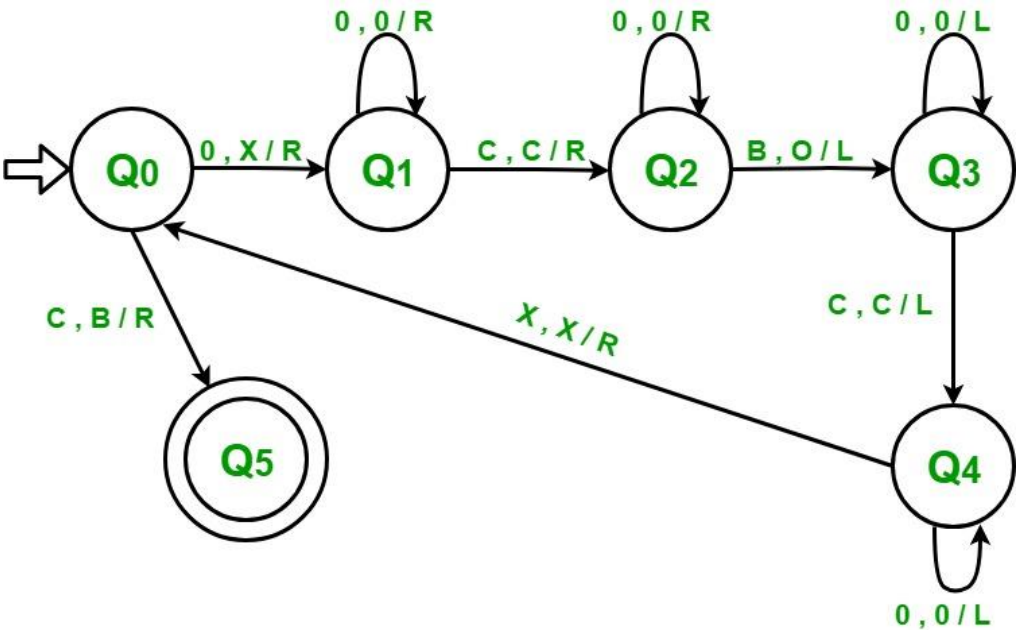
For adding 2 numbers using a Turing machine, both these numbers are given as input to the Turing machine separated by a "c".

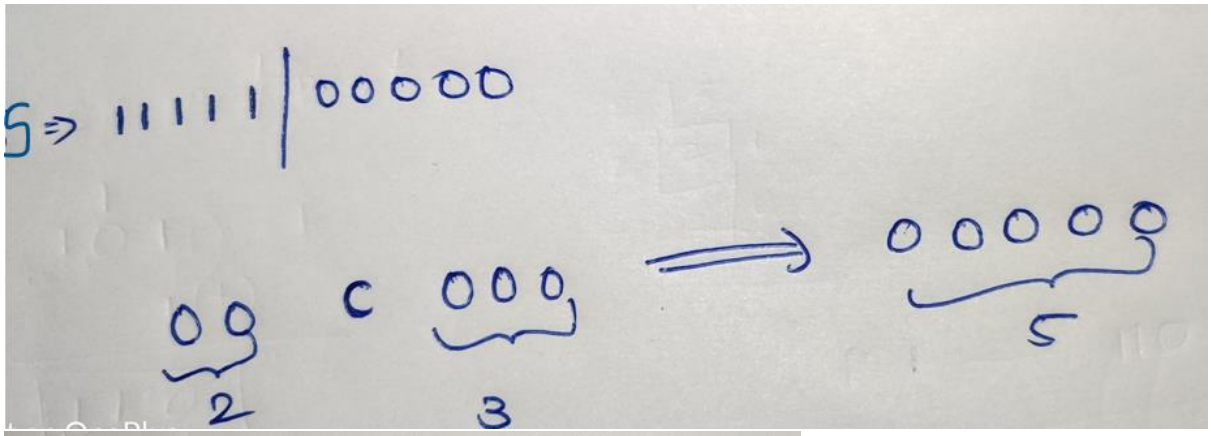
Examples – (2 + 3) will be given as 0 0 c 0 0 0:

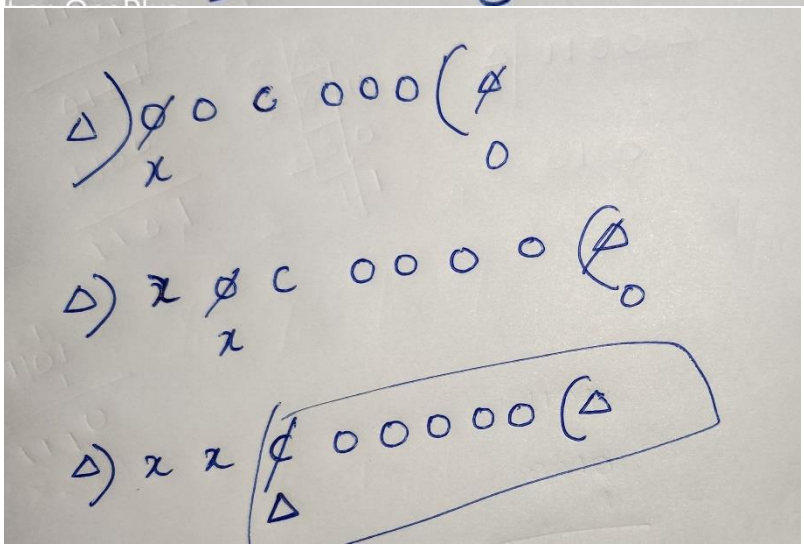
```
Input  :  0 0 c 0 0 0    // 2 + 3
Output :  0 0 0 0 0      // 5

Input  :  0 0 0 0 c 0 0 0  // 4 + 3
Output :  0 0 0 0 0 0 0    // 7
```

Approach used –

Convert a 0 in the first number in to X and then traverse entire input and convert the first blank encountered into 0. Then move towards left ignoring all 0's and "c". Come the position just next to X and then repeat the same procedure till the time we get a "c" instead of X on returning. Convert the c into blank and addition is completed.




5 ⇒ 11111 | 00000
00 c 000
2
3
00000
5


Δ) x 0 c 000 (x 0
Δ) x x c 0000 (0
Δ) x x x 00000 (Δ Δ

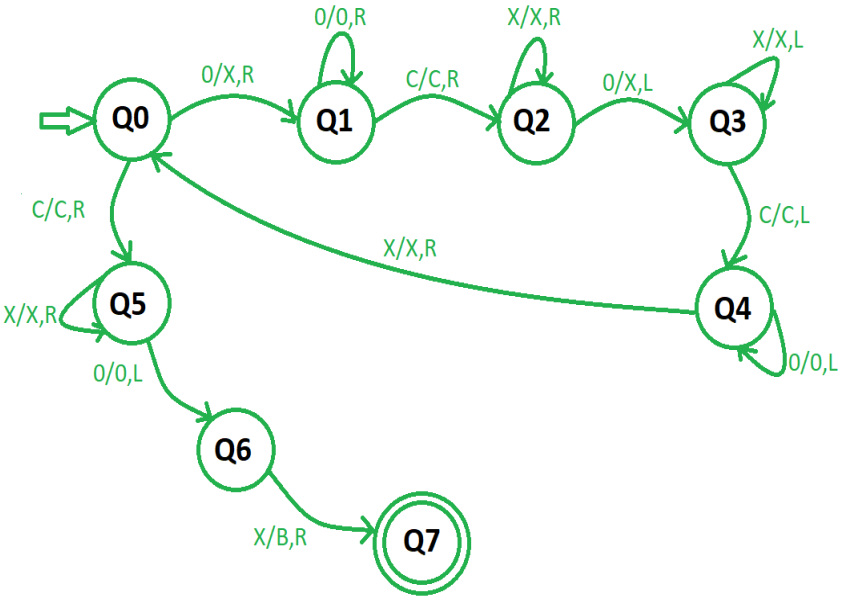
Turing machine for subtraction ***(Unary representation of a number)***

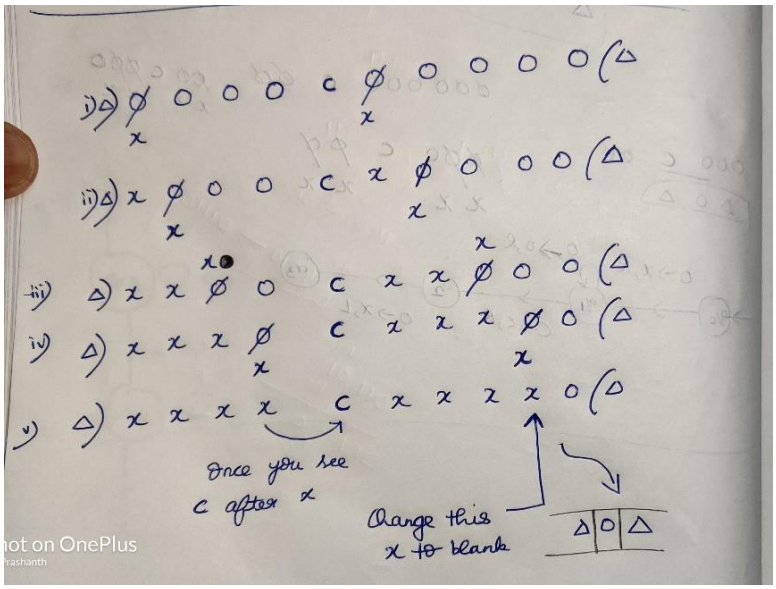
Example – (3 – 4) or (4 – 3) will be given as 0 0 0 c 0 0 0 0

```
Input:  0 0 0 c 0 0 0 0  // (3 - 4) or (4 - 3)
Output:  0               // (1)
```

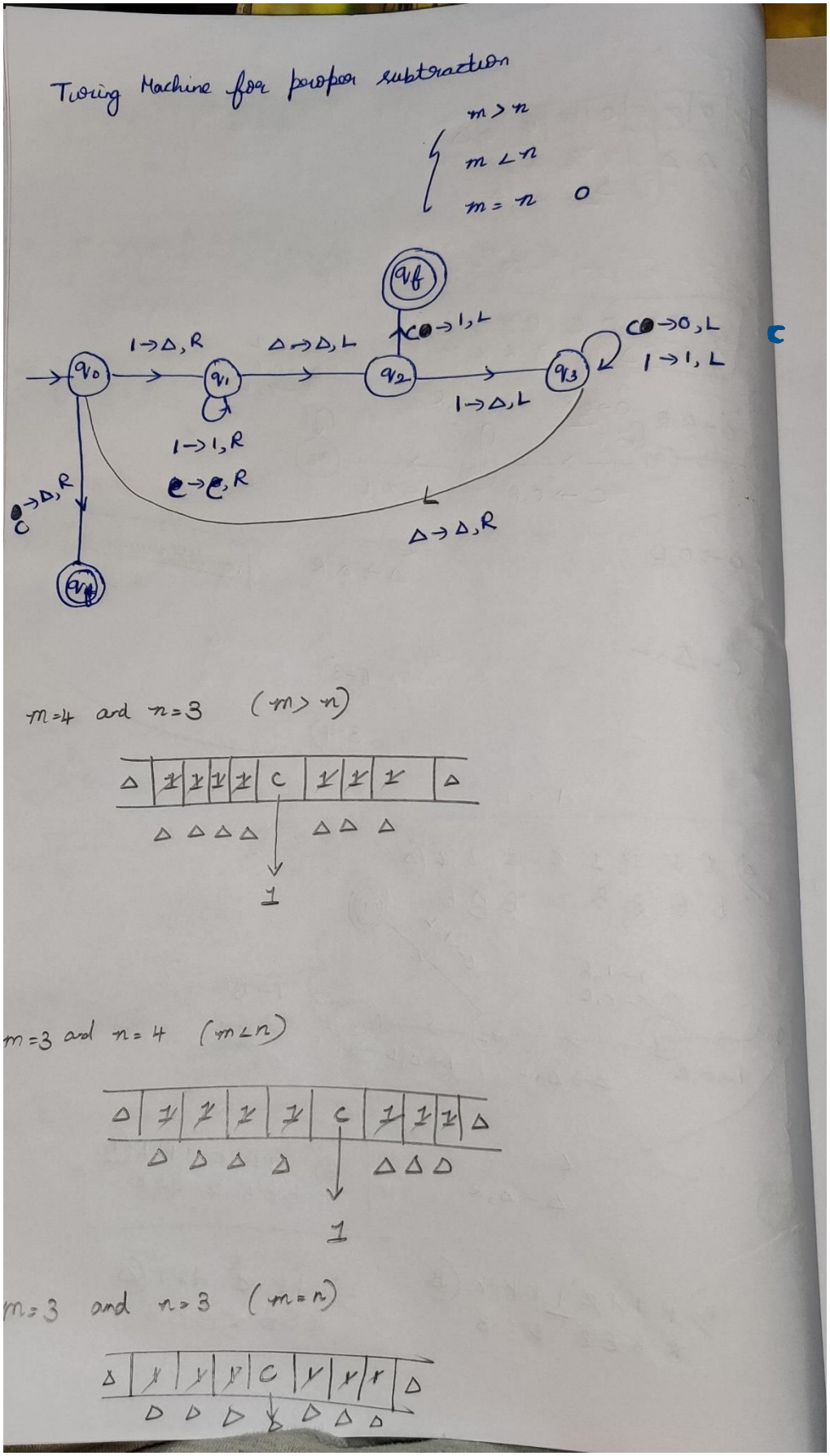
Approach used –

Convert a 0 in the first number into X and then move to the right, keep ignoring 0's and "c" then make the first 0 as X and move to the left . Now keep ignoring 0's, X's and "c" and after finding the second zero repeat the same procedure till all the zeros on the left hand side becomes X .Now move right and convert the last X encountered into B(Blank).

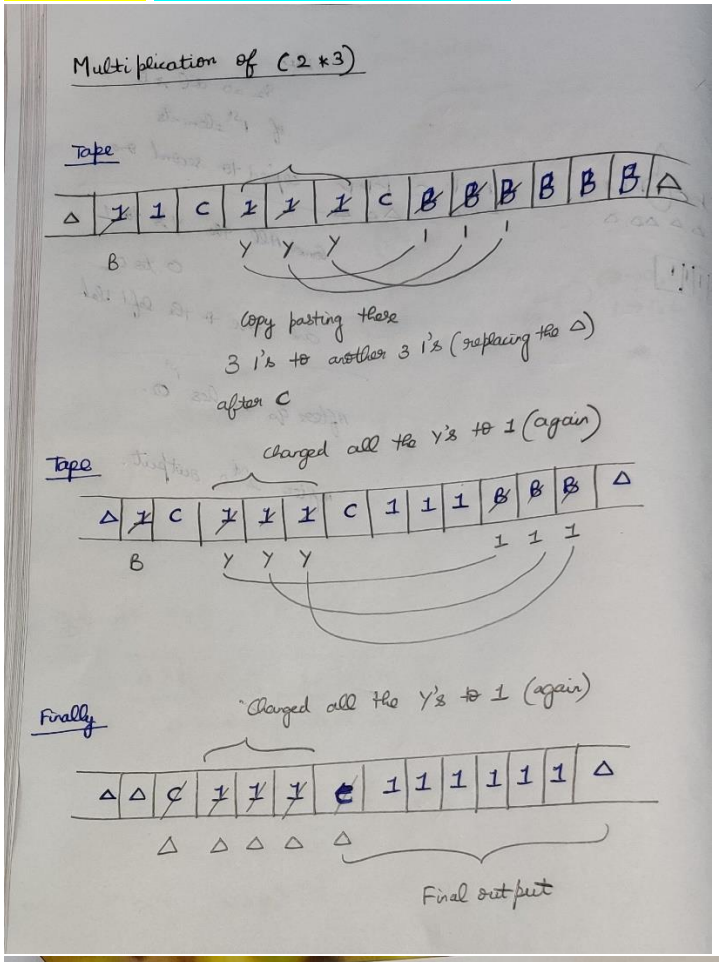


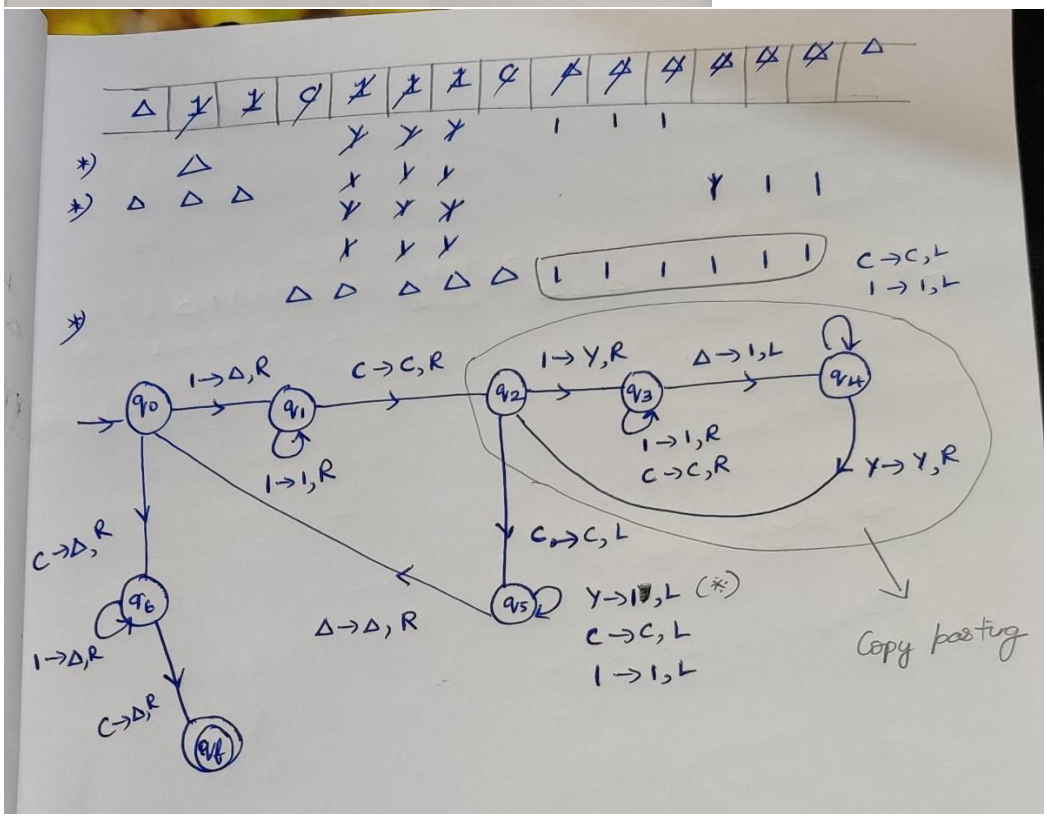


Question (Proper Subtraction)

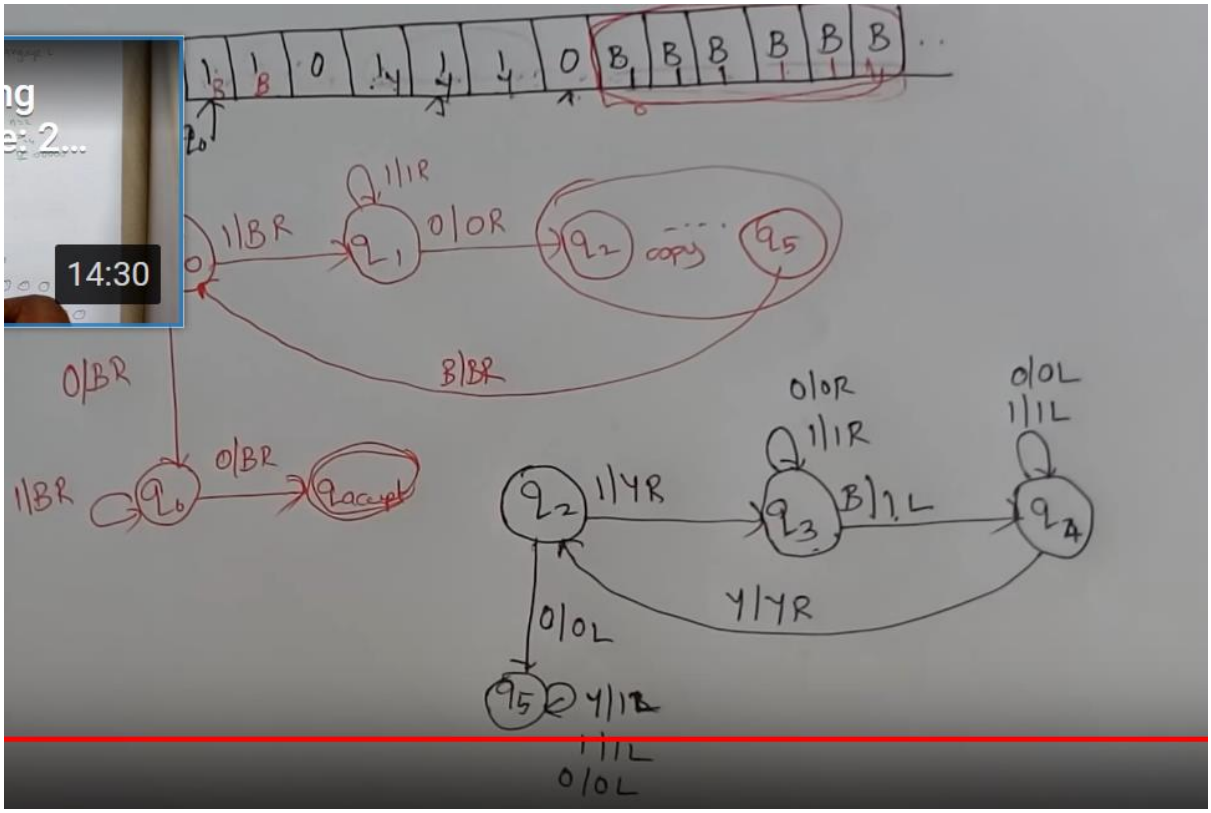


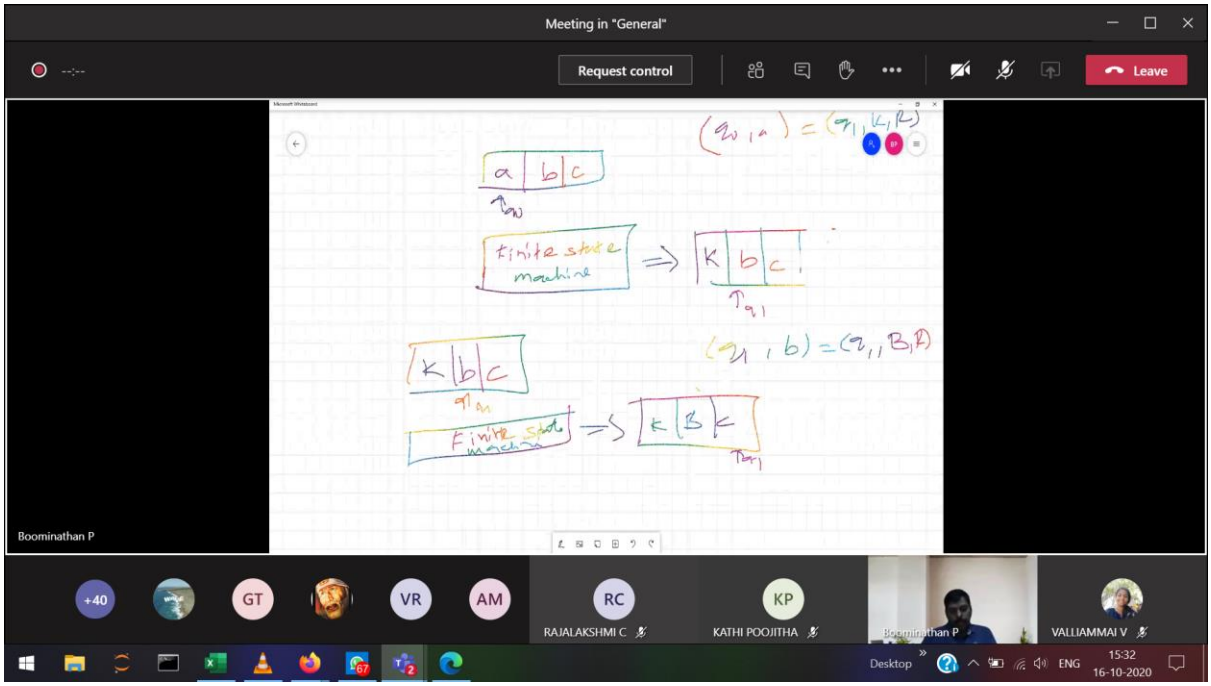
Question (Proper Multiplication) [2*3] and [3*2] separately

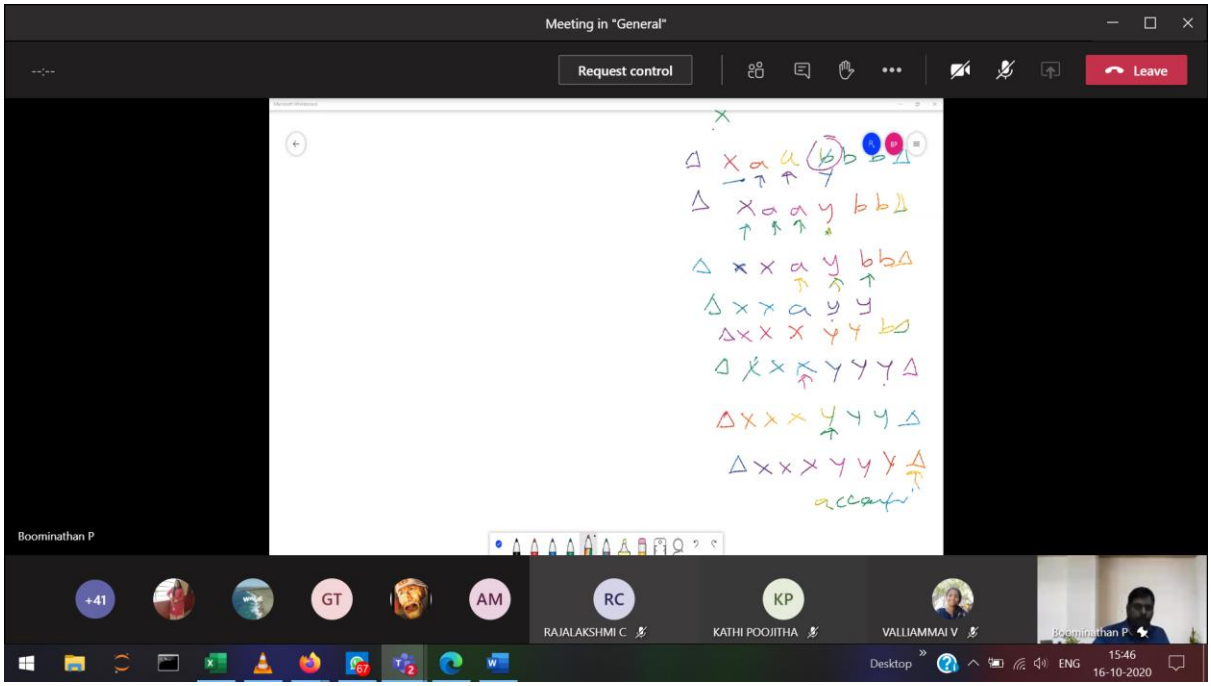




Question (Proper Multiplication) [Subroutines]






Meeting in "General"
Request control
Leave
Boominathan P
+41
GT
AM
RC
KP
RAJALAKSHMI C
KATHI POOJITHA
VALLIAMMAI V
Boominathan P
Desktop
ENG
15:46
16-10-2020

Instantaneous Description