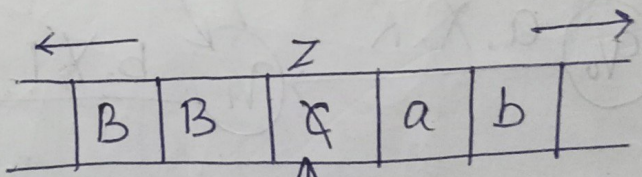
Turing Machine :

Turing machine is a mathmatical model of computation that define abstract computer. It was invented in 1936 by Alan Turing.

Following is the mathmatical model of turing machine.

$$T.M = \{Q, \Sigma, q_0, F, \delta, \Gamma, B, \delta\}$$



Γ = Toe Alphabet
[Set of all alphabet]

$\Sigma \subseteq \Gamma$

B = Blank Symbol

R/W Reader

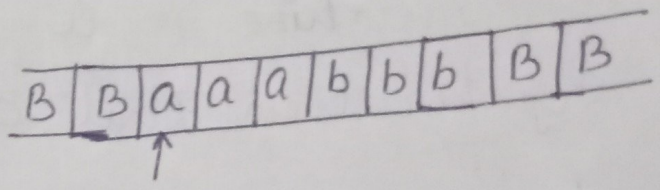
Finite Control Unit

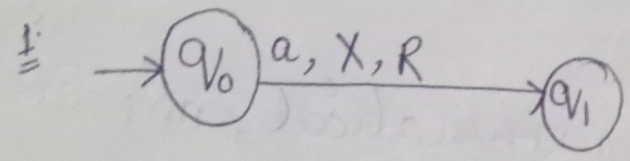
* Write is mandatory

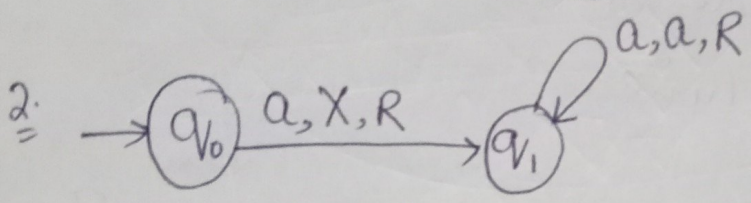
$$\delta \rightarrow Q \times \Gamma \longrightarrow Q \times \Gamma \times L/R$$

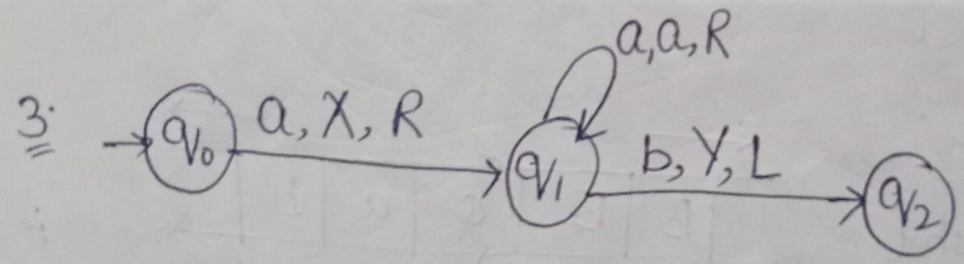
(Read + write) (Read, write, skip)

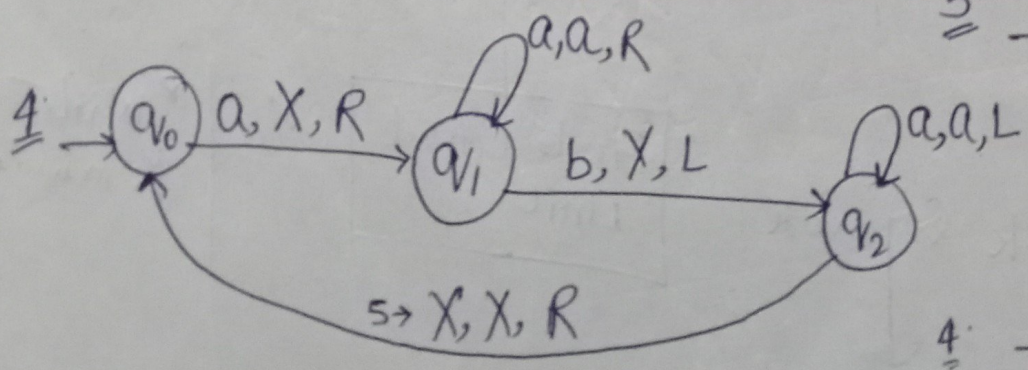
$L = \{ a^n b^n / n \geq 1 \}$

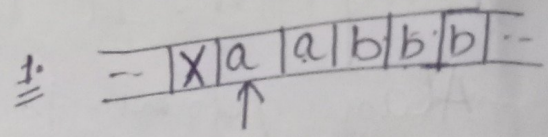

B B a a a b b b B B

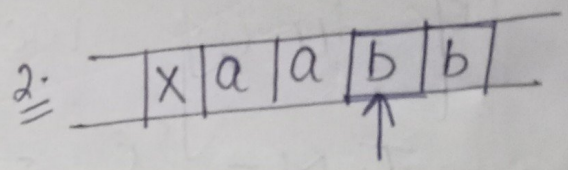

1.
q_0
a, X, R
q_1

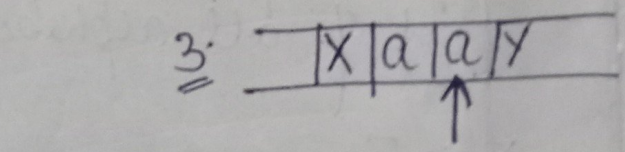

2.
q_0
a, X, R
q_1
a, a, R

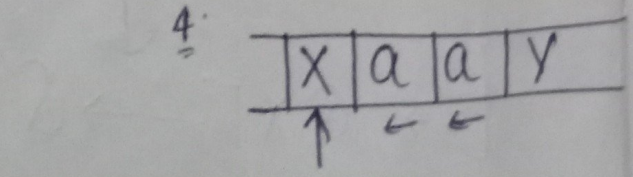

3.
q_0
a, X, R
q_1
a, a, R
b, Y, L
q_2

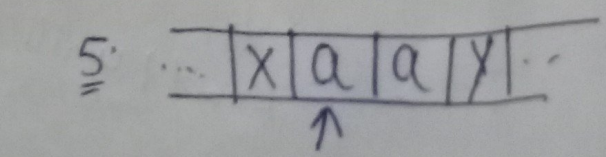

4.
q_0
a, X, R
q_1
a, a, R
b, Y, L
q_2
a, a, L
5→ X, X, R


1.
X a a b b b

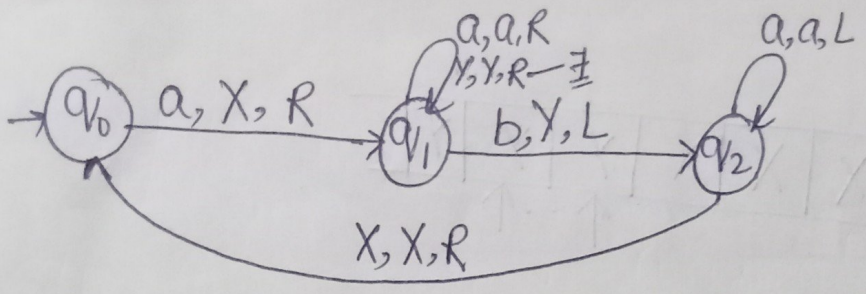

2.
X a a b b


3.
X a a Y

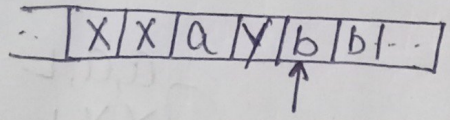

4.
X a a Y


5.
X a a Y

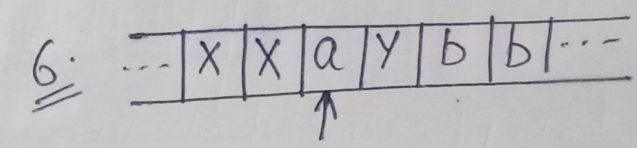
6.


a, a, R
Y, Y, R
a, a, L
q_0
a, X, R
q_1
b, Y, L
q_2
X, X, R

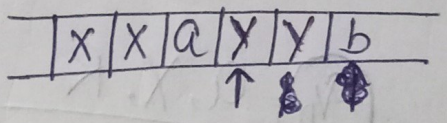
7.

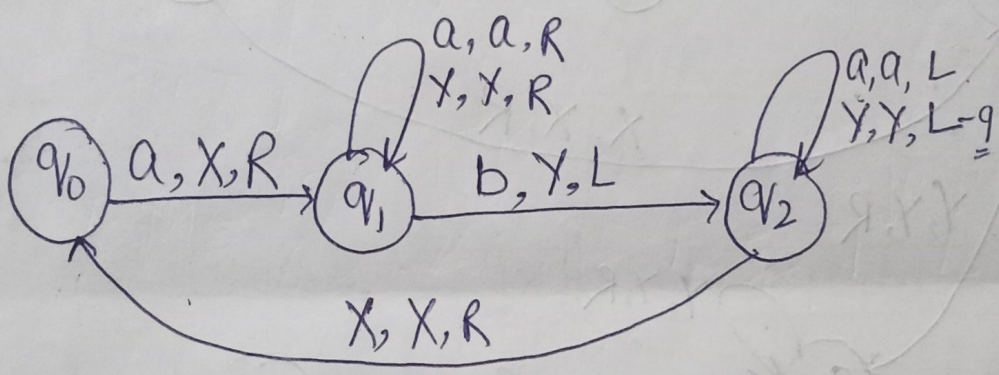

X X a Y b b

8.

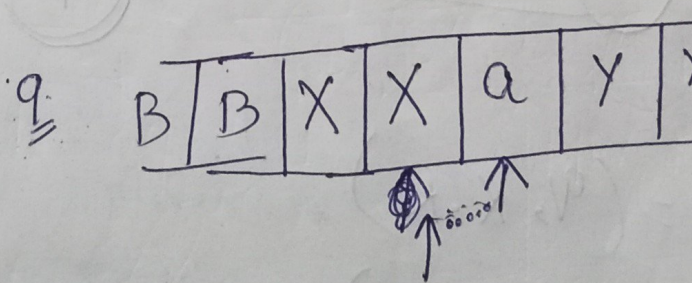

X X a Y b b

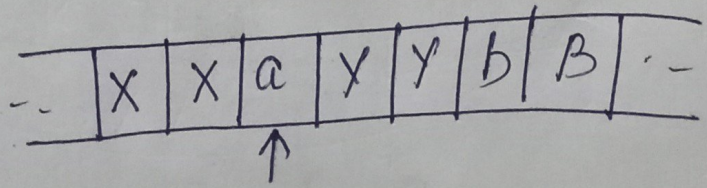
9.


X X a Y Y b

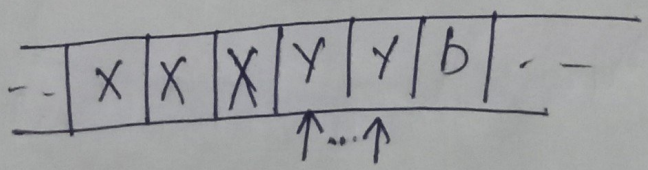

a, a, R
Y, Y, R
a, a, L
Y, Y, L
q_0
a, X, R
q_1
b, Y, L
q_2
X, X, R

10.

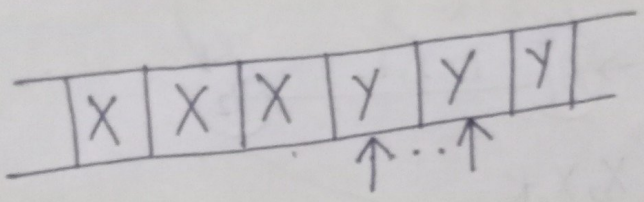

B B X X a Y Y b B


X X a Y Y b B

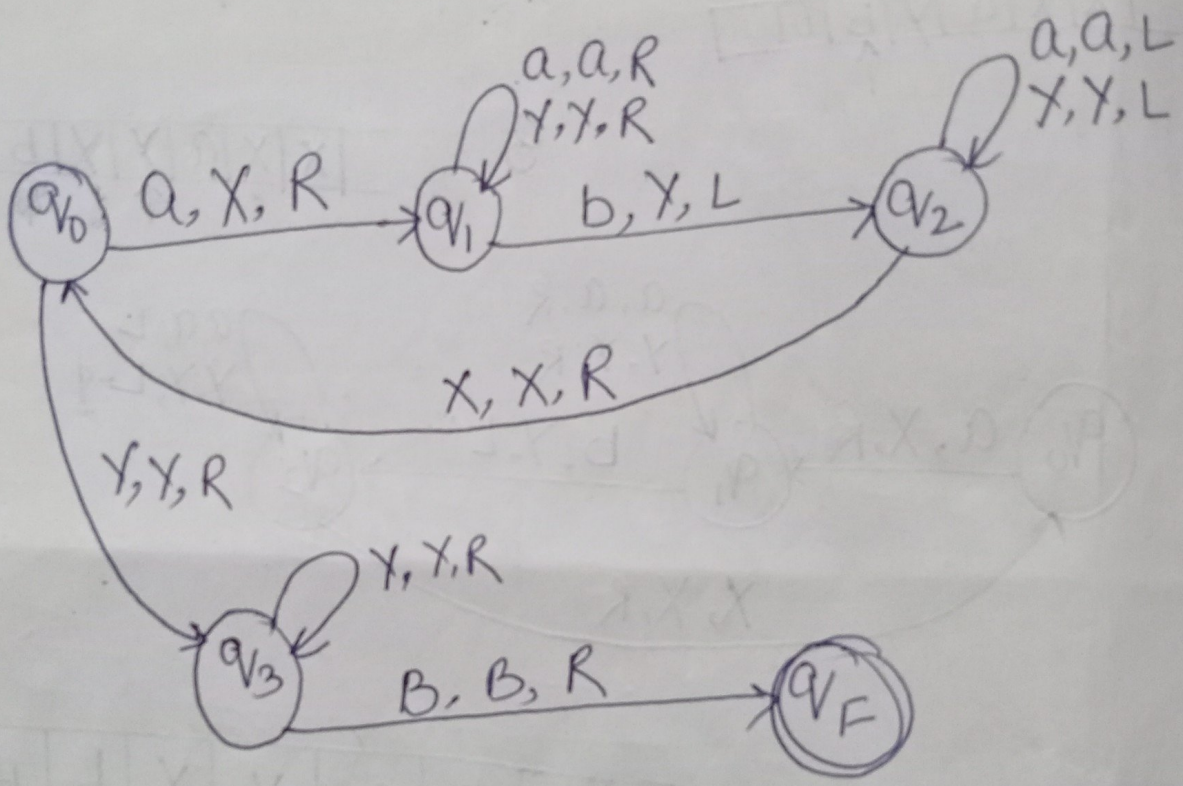
11.


X X X Y Y b

12



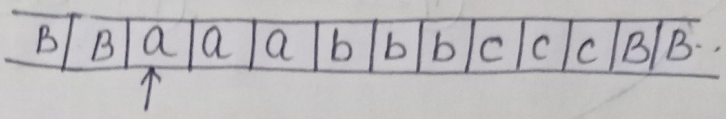
(13)

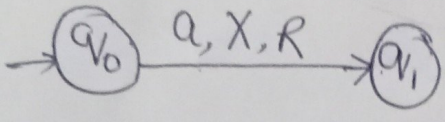


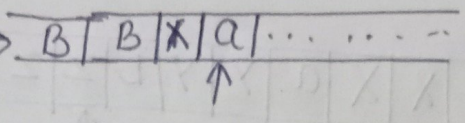
$\delta: (q_0, a) = (q, X, R)$

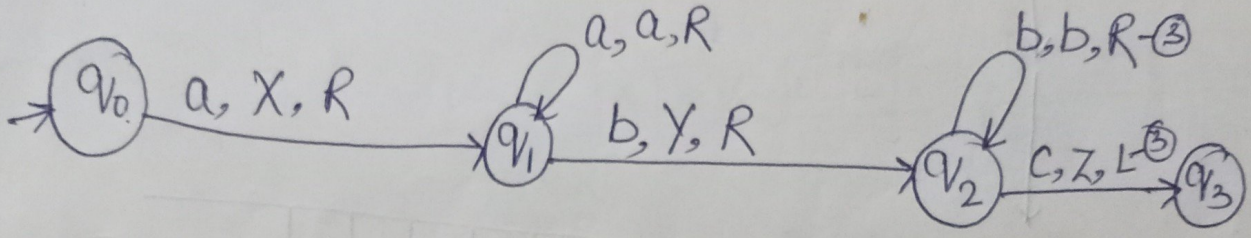
$(q_1, b) = (q_2, Y, L)$

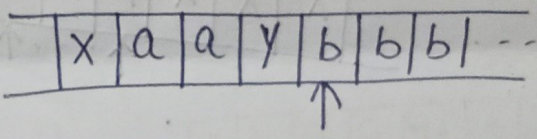
$L = \{a^n b^n c^n\}, n \geq 1$

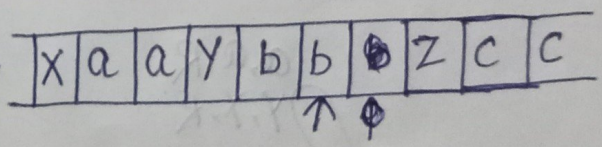


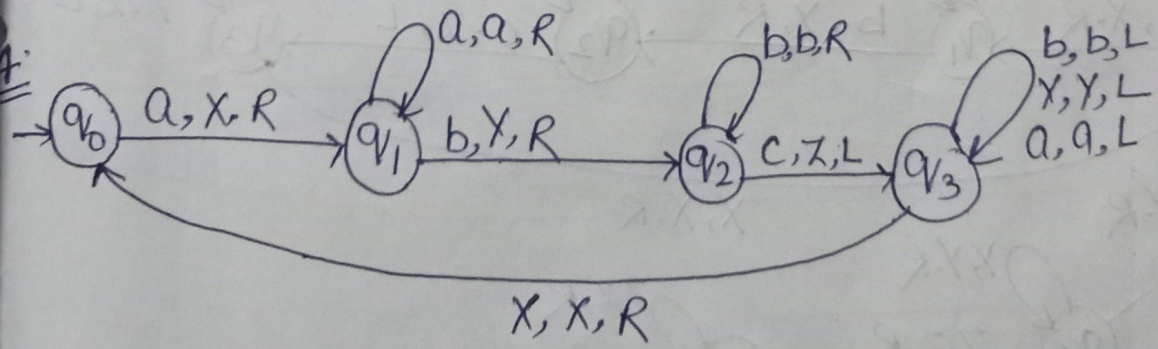
1. 

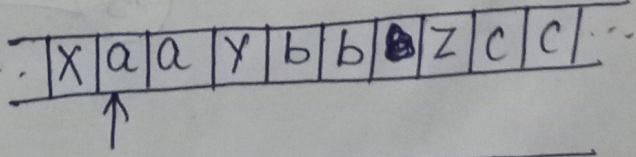
1 → 

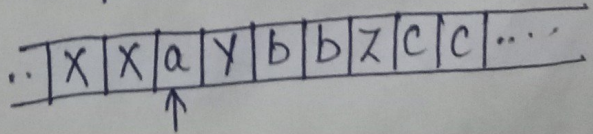
2. 

2. 

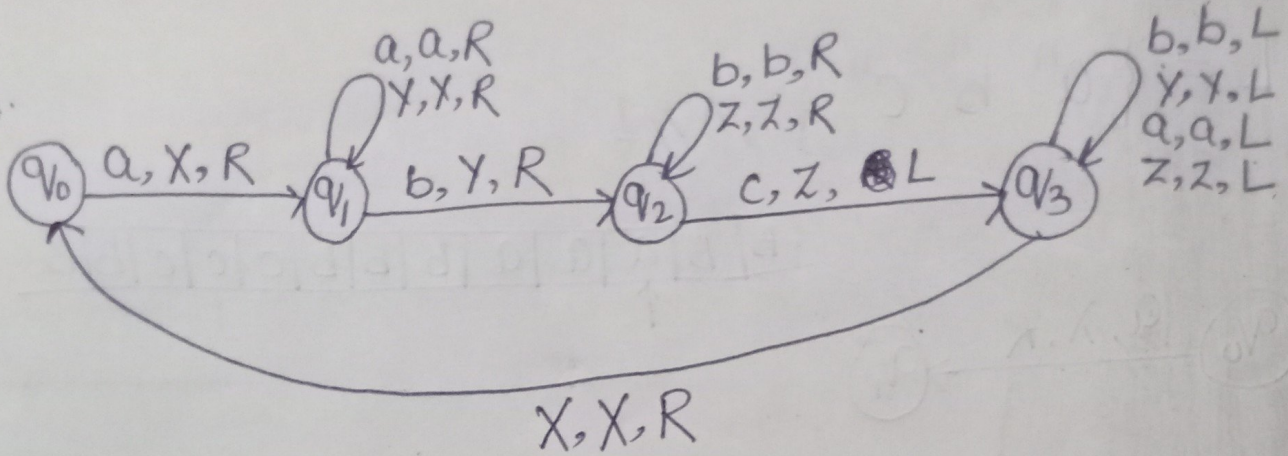
3. 

4. 

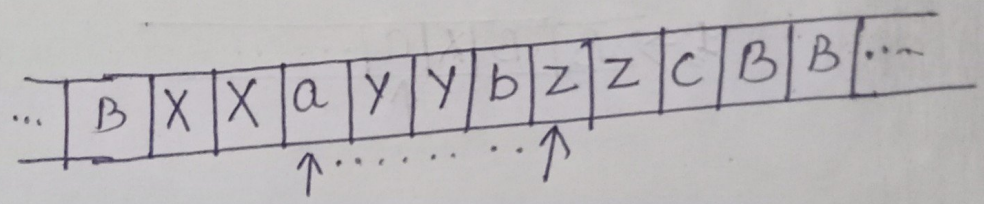
4 → 

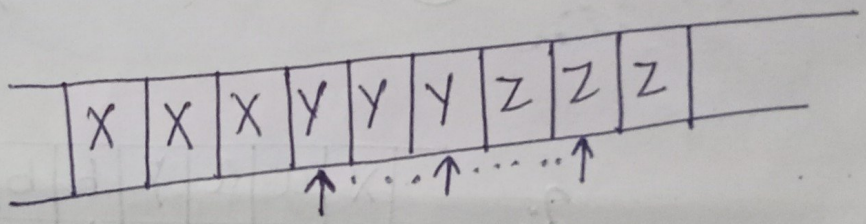
5. 

5.

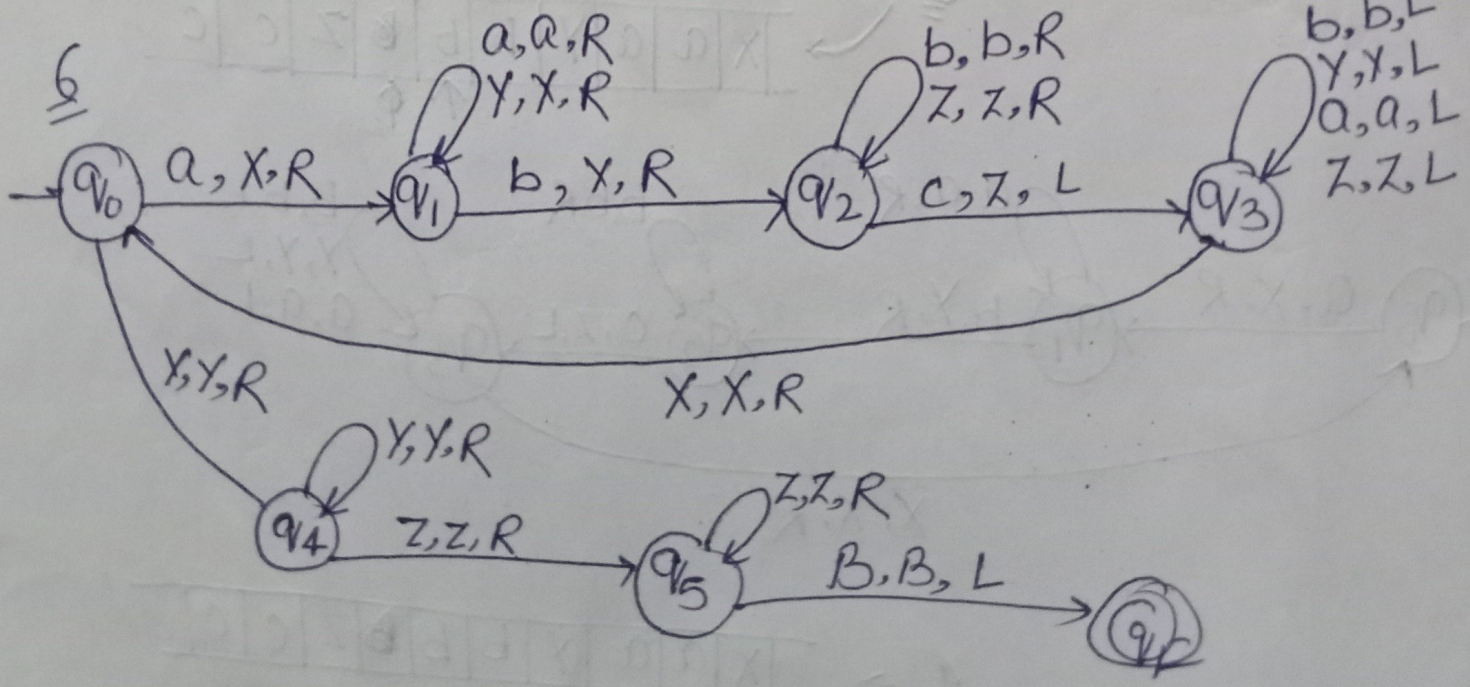

a, a, R
Y, X, R
b, b, R
Z, Z, R
b, b, L
Y, Y, L
a, a, L
Z, Z, L
q0
a, X, R
q1
b, Y, R
q2
c, Z, L
q3
X, X, R

5→


B X X a Y Y b Z Z C B B

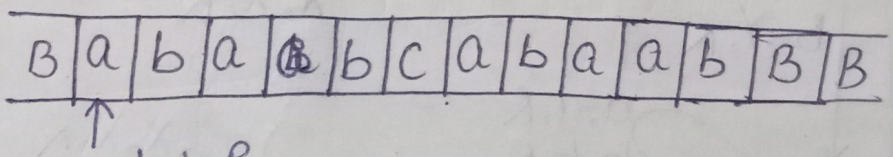

X X X Y Y Y Z Z Z

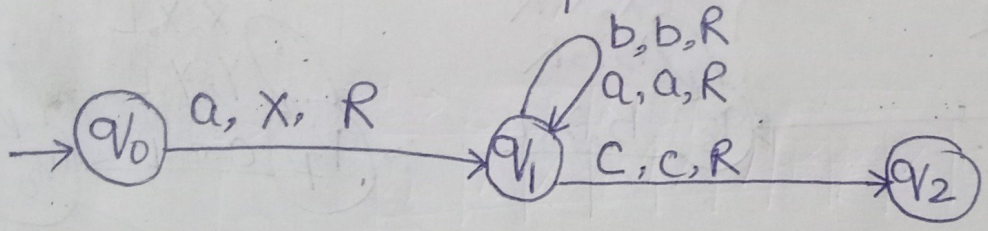
6


a, a, R
Y, X, R
b, b, R
Z, Z, R
b, b, L
Y, Y, L
a, a, L
Z, Z, L
q0
a, X, R
q1
b, X, R
q2
c, Z, L
q3
X, X, R
Y, Y, R
Y, Y, R
q4
Z, Z, R
Z, Z, R
q5
B, B, L
qf

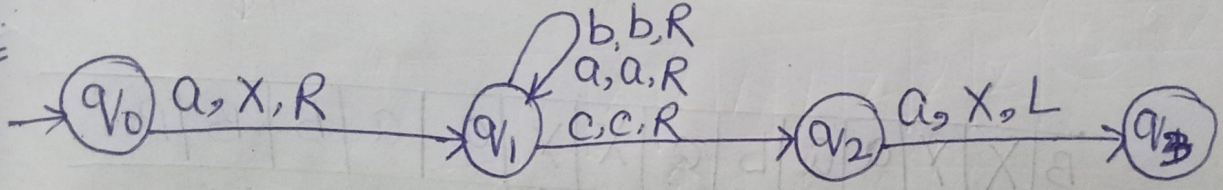
$L = \{ wcw / w \in \{a,b\}^* \}$

Solution:

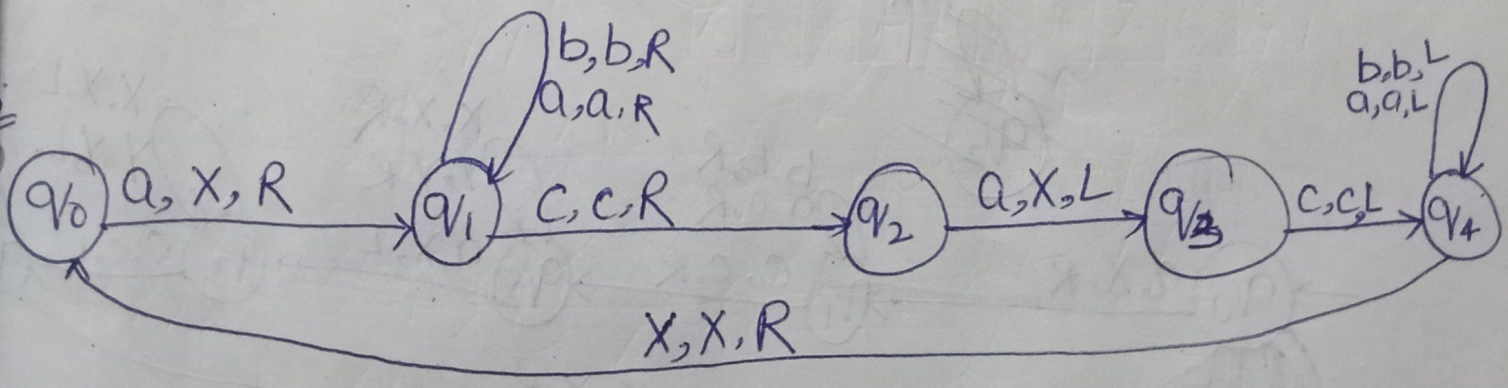




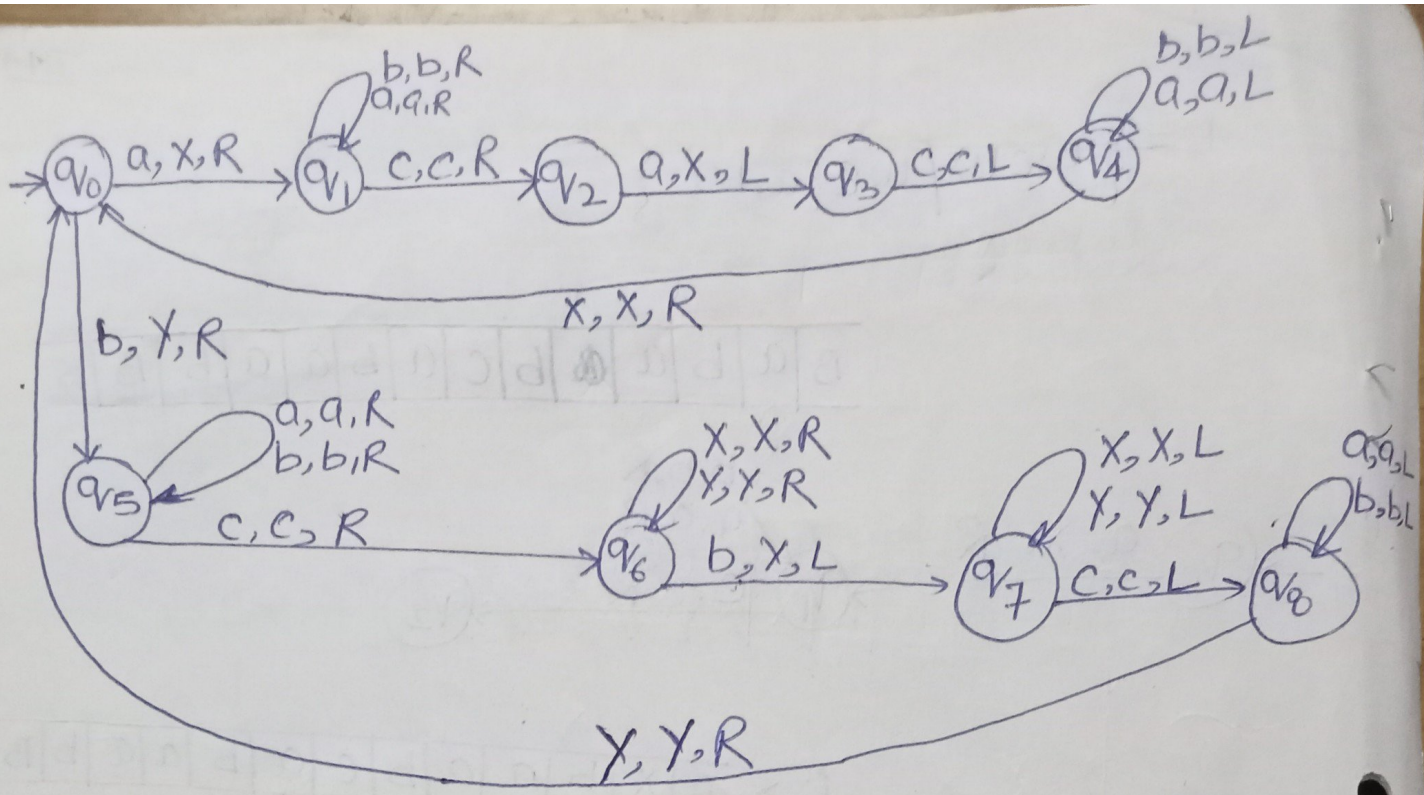
1 → B X b a a b c a b a a b B

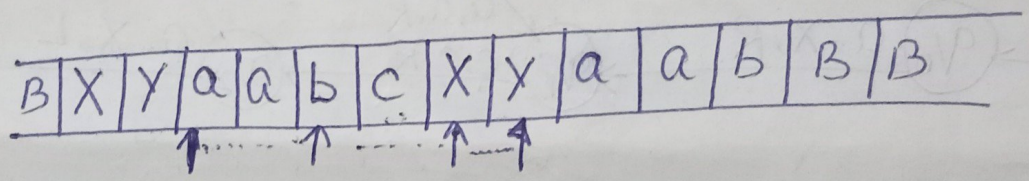


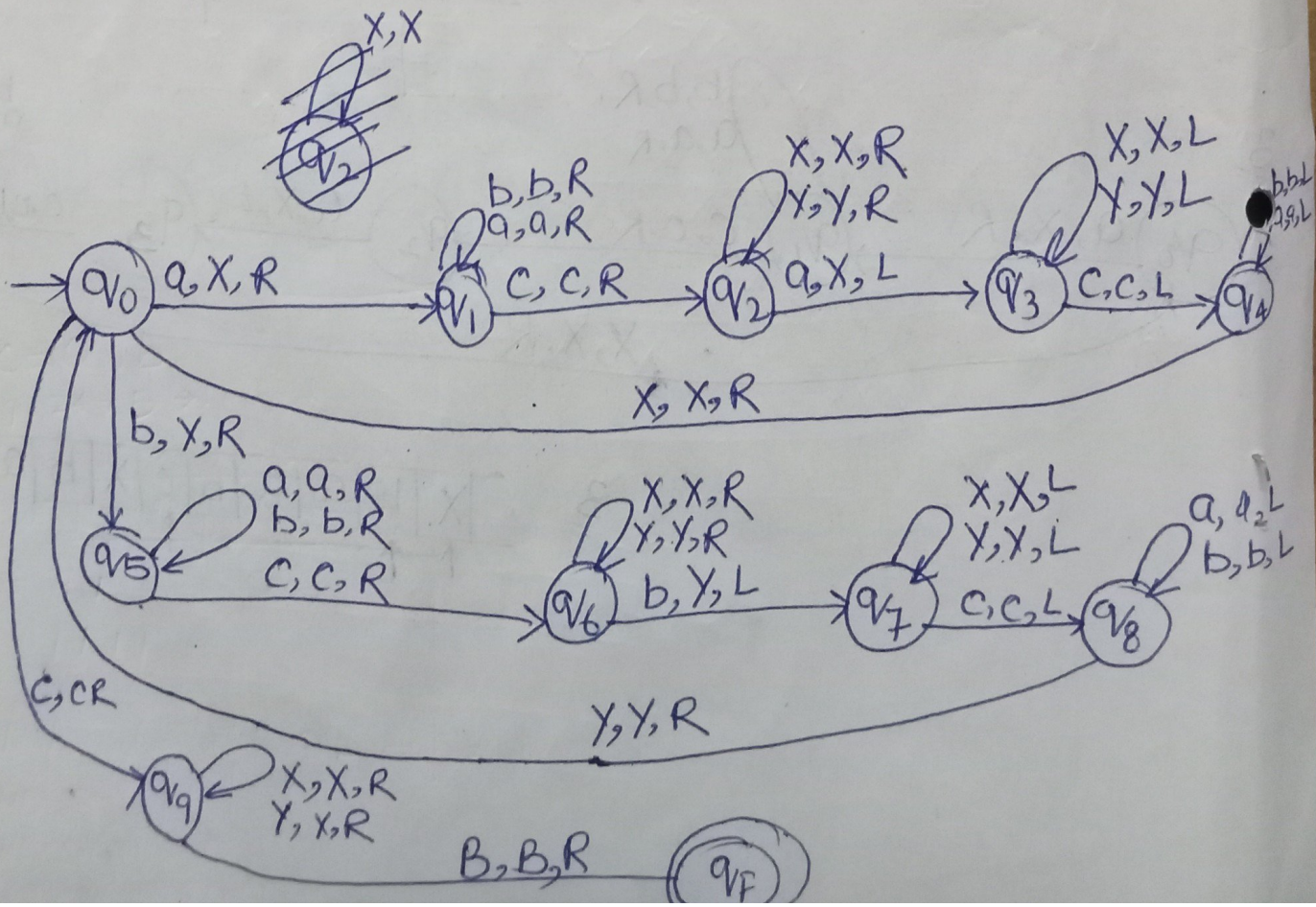
2 → X b a a b c X b a a b B



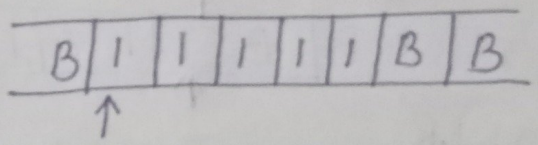
3 → X b a a b c X b a a b B


q0
q1
q2
q3
q4
q5
q6
q7
q8
a, X, R
b, b, R
a, a, R
c, c, R
a, X, L
c, c, L
b, b, L
a, a, L
X, X, R
b, Y, R
a, a, R
b, b, R
c, c, R
X, X, R
Y, Y, R
b, X, L
X, X, L
Y, Y, L
c, c, L
a, a, L
b, b, L
Y, Y, R


B X Y a a b c X Y a a b B B


X, X
q2
b, b, R
a, a, R
X, X, R
Y, Y, R
X, X, L
Y, Y, L
b, b, L
a, a, L
q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
qF
a, X, R
c, c, R
a, X, L
c, c, L
X, X, R
b, X, R
a, a, R
b, b, R
c, c, R
b, Y, L
c, c, L
a, a, L
b, b, L
Y, Y, R
c, c, R
X, X, R
Y, X, R
B, B, R

Turning Machine as unary to binary Converter:



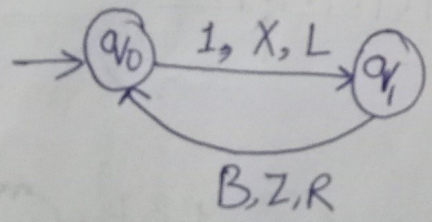
binary = 101
ZYZ

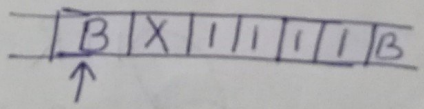
1-X
0→Y
1→Z

U	B
1	1
11	10
111	11
1111	100

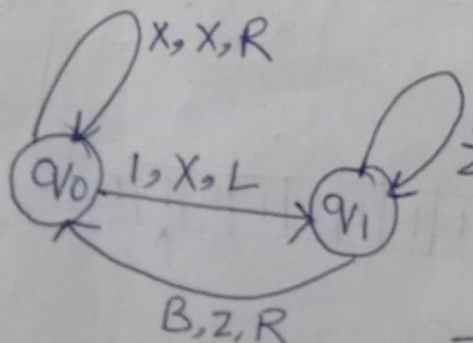
Binary ⇧ add

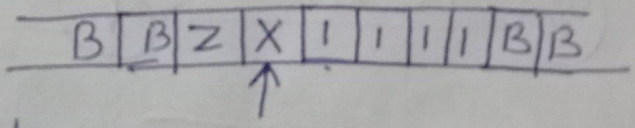
1:

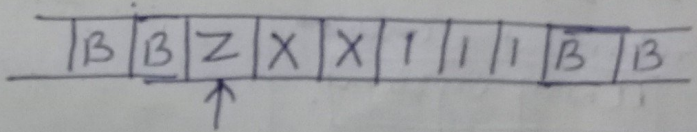




2:



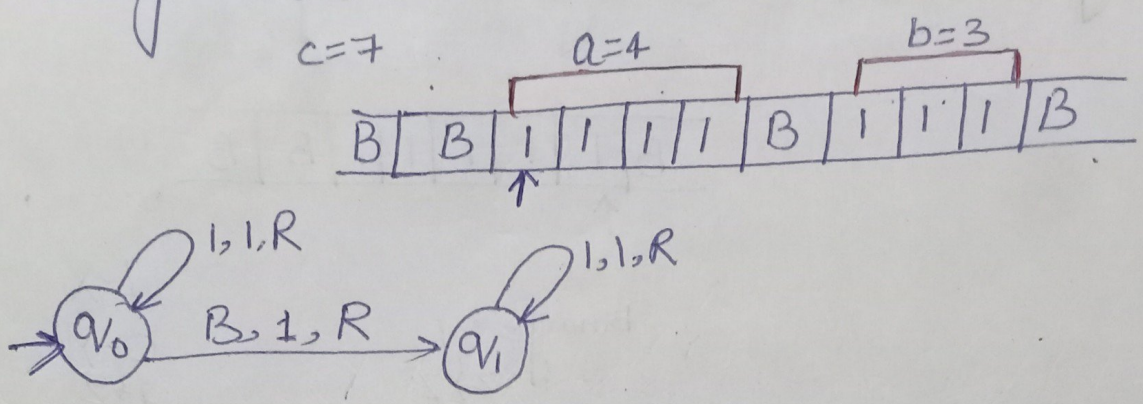


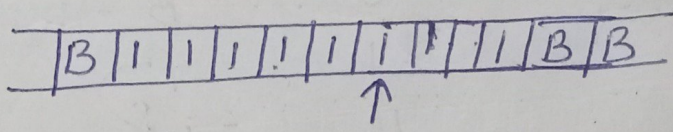


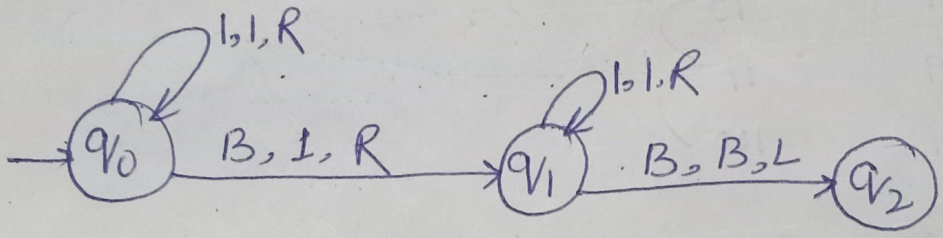
$$\begin{array}{r} 1\,Z \\ +\,1\,Z \\ \hline 10 \\ ZY \end{array}$$

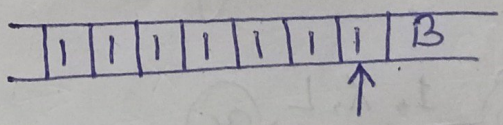
Unary System

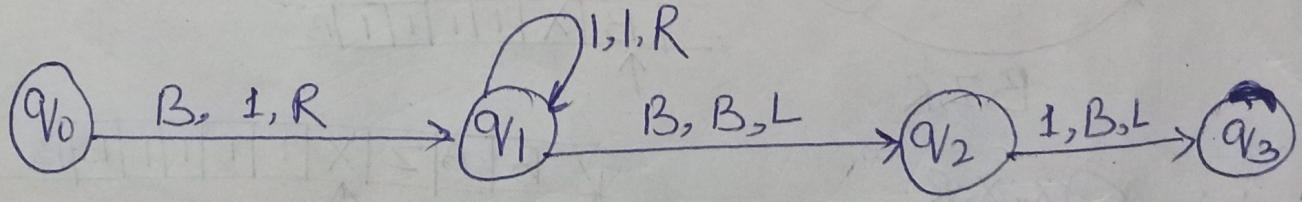
Turning Machine as adder :

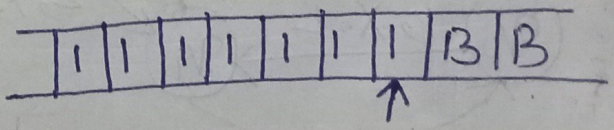












$c = a + b$

29 24+5 → Decimal

Lets represent all no's into Unary System.

2 = 11

4 = 1111

* No final state coz
* we are performing function not a string is accepted

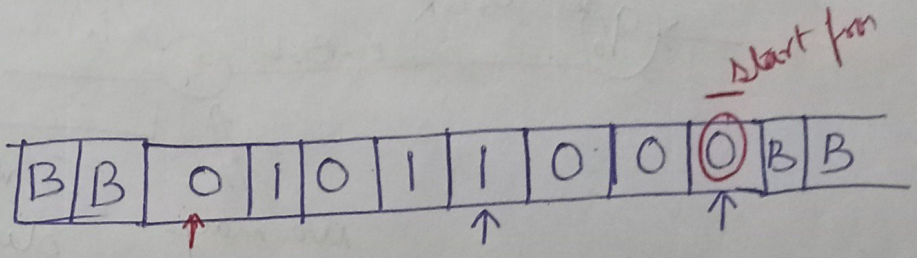
TM for 2's compliment :

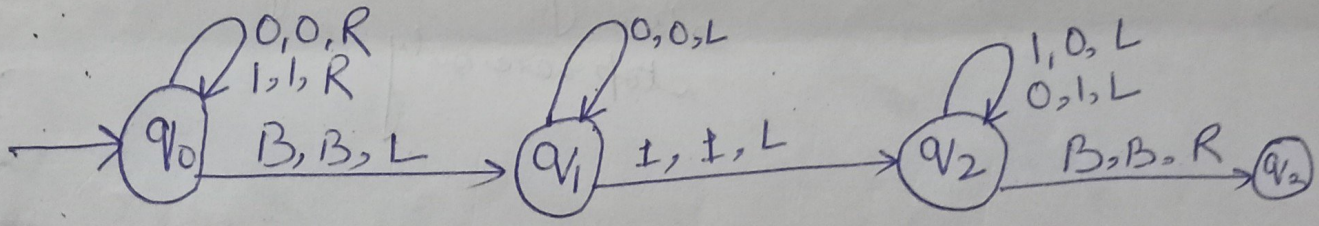
0 1 0 1 1 0 0 0

1 0 1 0 0 1 1 1 ⟶ 1's compliment

+ 1

1 0 1 0 1 0 0 0 ⟶ 2's complement





✓

**Observation:

0 1 0 1 1 0 0 0

1 0 1 0 1 0 0 0

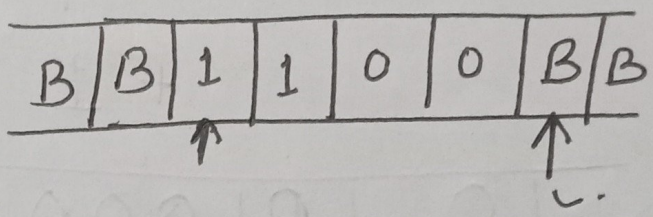
Reverse

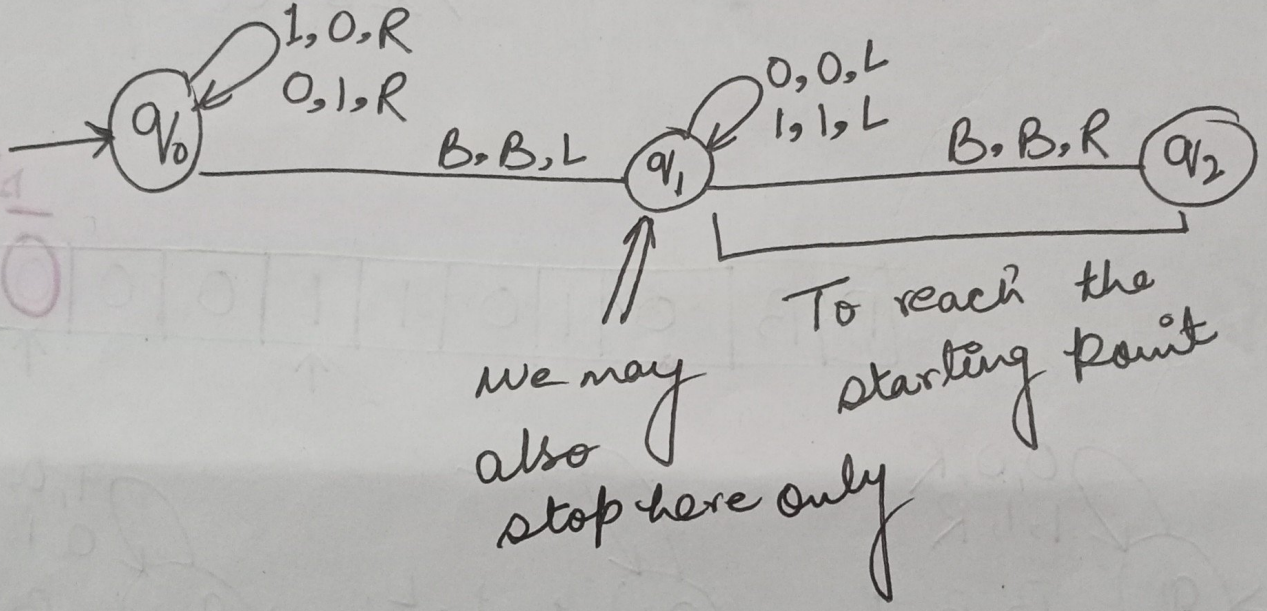
*Once first 1 encountered all remaining no's changed

Turing Machine for 1's Complement :

w = 0 1 0 1 0 0

1's C = 1 0 1 0 1 1





The imitation game 2014

Church-turing thesis: A function on natural number is computable by an algo if and only if it is computable by turing machine (1936)

→ ~~Algo is Turing~~ Anything that can be done by current digital computer can also be done by a turing machine.

* Predictable Algo

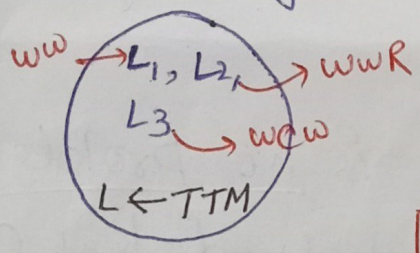
→ currently there is no problem which can be solved by a digital computer and can not be solved by a TM.

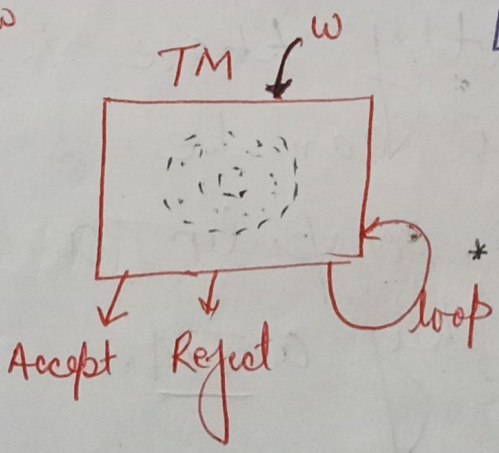
→ many mathmatical model are suggested but no one of them is more powerful then the turing machine.

Recursively Enumerable:

A Language L is said to be recursively enumerable, if there exists a turing machine that accepts it.

This definition implies only that there exists a turing machine M, such that, for every $w \in L$





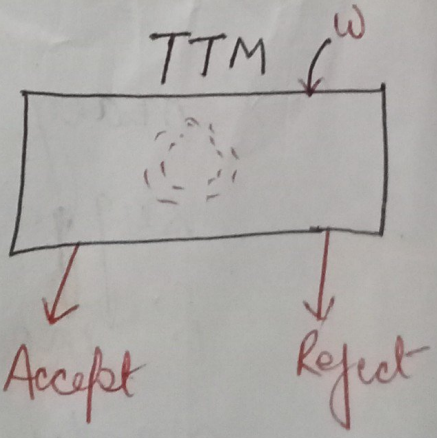
L → Valid Input

$w \in L$ → Accept

$w \notin L$ → Reject/Loop

↳ invalid Input

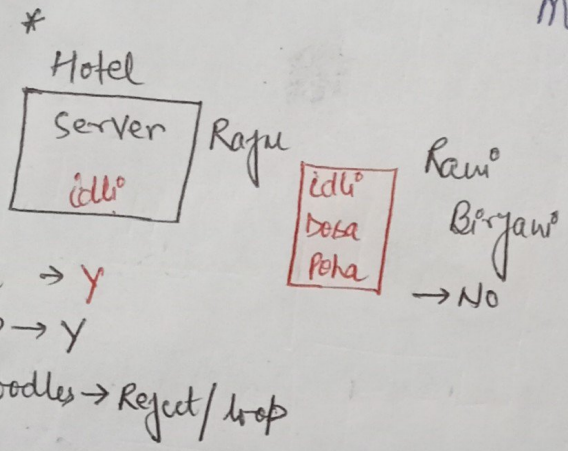
$$\begin{bmatrix} L \\ w \in L \\ w \notin L \end{bmatrix}$$



No looping in total turing machine

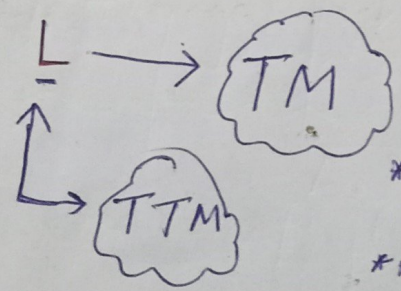
Recursive Language:

A language L is said to be recursive, if there exists a total turing machine (TTM)



* Well behaved

Q1:

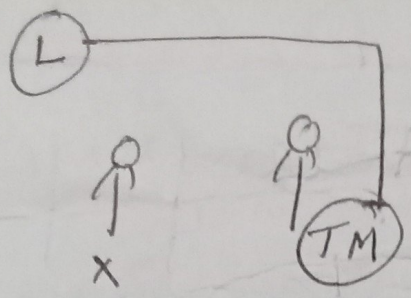


* i dont know whether it is TTM

** Can we have TTM for every Language

not compulsory that every Language have TTM ***

$$L \overset{X}{\Rightarrow} TM \Rightarrow TTM?$$



If some one construct

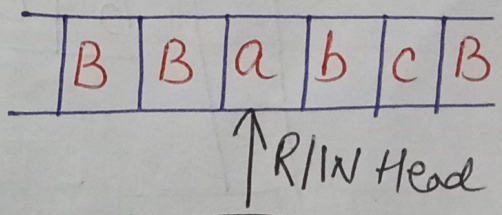
Variations of Turing Machine:

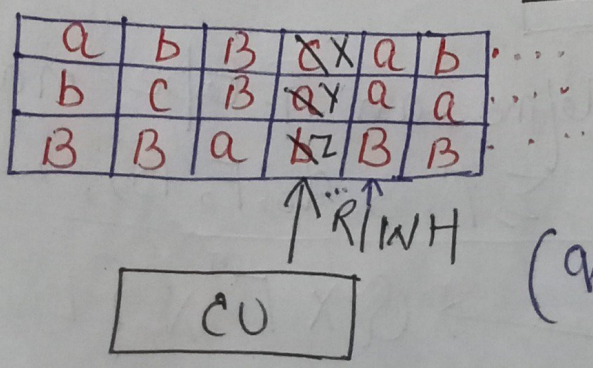
(1) Multitrack TM
(2) Multitape ,,
(3) 2- stack Machine -XX
(4) Non- deterministic TM

[1] Multitrack Turing Machine:

A MTTM is one in which the tape can be viewed as extending infinitely in more than one dimension/track. Following diagram shows the concept.

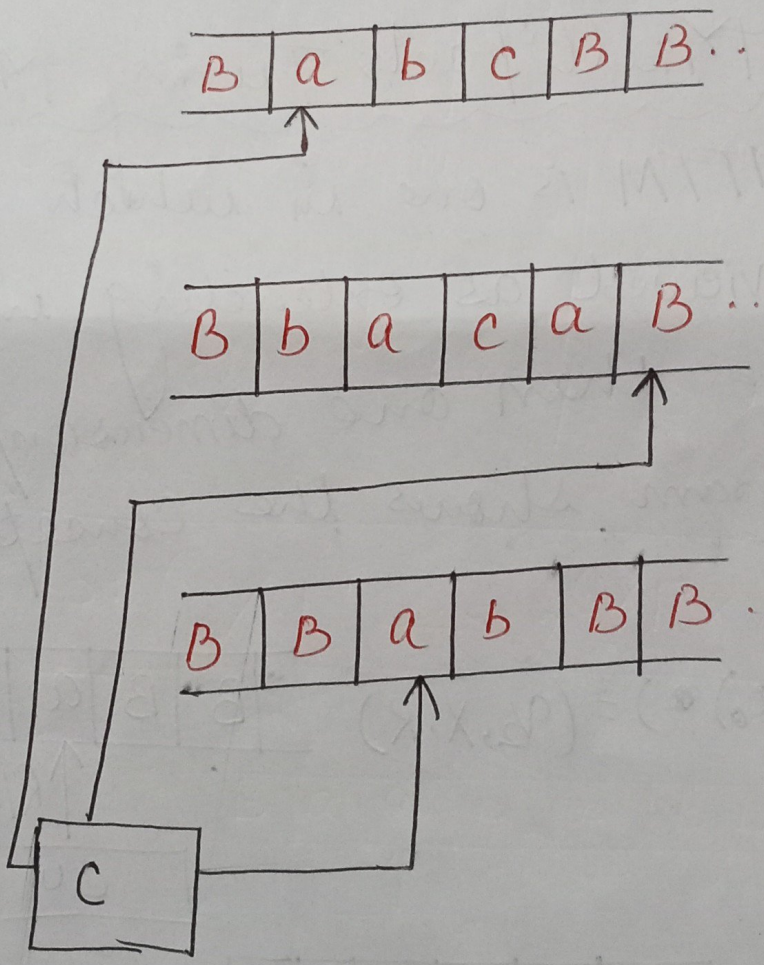
$(q, a_0) = (q_2, X, R)$





$(q, c, a, b) = (q_{next}, X, Y, Z, R)$

[2] Multitape Turing Machine : A multitape turing machine is a turing machine with several tapes, each of ~~which~~ with its own independently controlled read-write head Following fig shows the concept.



Typically, we define an n-tape machine by $M = \{Q, \Sigma, \Gamma, \delta, q_0, F, B\}$, where

$$\delta: Q \times \Gamma^n \longrightarrow Q \times \Gamma^n \times \{L, R\}^n$$

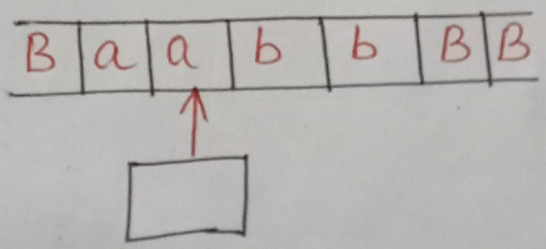
specifies what happens on all the tapes.
For example, if $n=3$ with a current configuration shown in the above fig, then

$$\delta(q_0, a, B, a) = (q_1, X, Y, X, L, R, L)$$

Nondeterministic Turing Machine :

A nondeterministic turing machine is an automata as given by the definition of turing machine, except δ is now a function.

$$\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$$



DTM $\longrightarrow (q, a) = (q, X, L)$

NDTM $\longrightarrow (q, a) = (q_1, X, L)$
(q_2, Y, R)
(q_3, Z, L)
⋮ ⋮ ⋮

As always when nondeterminism is involved, the range of δ is a set of possible transactions, any of which can be chosen by the machine.

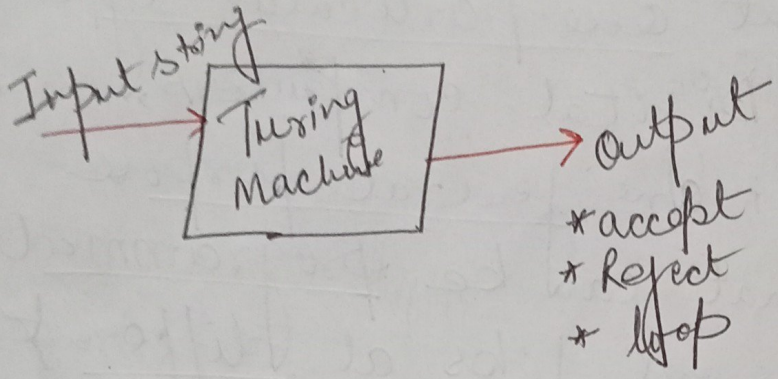
Universal Turing Machine: Consider the following argument against the Turing's thesis:

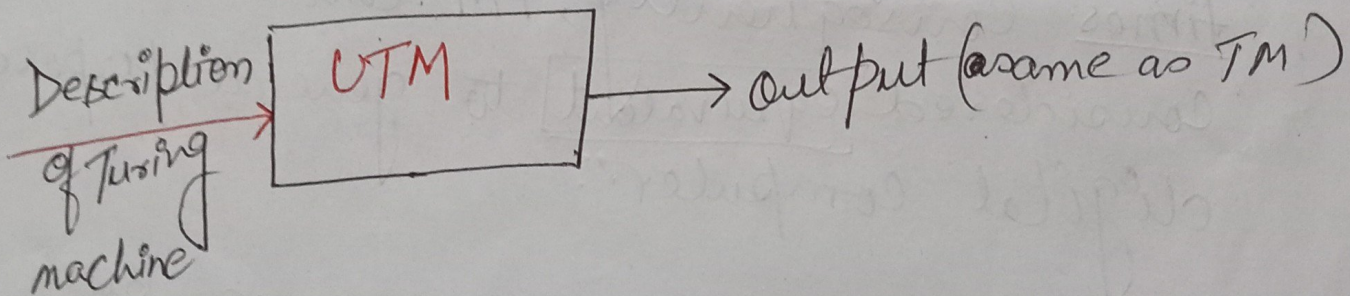
"A TM is a special purpose computer. Once δ is defined, the machine is restricted to carrying out one particular type of computation. Digital computers, on the other hand, are general purpose machines that can be programmed to do different jobs at different times. Consequently, TM can not be considered equivalent to general purpose digital computer".

This objection can be overcome by designing a reprogrammable turing machine, called a universal Turing machine.

A UTM, M_U is an automaton that, given as input the description of any turing machine M and a string w,

can simulate the computation of M on w. To construct such an M_u, we first chose a standard way of describing Turing machines.

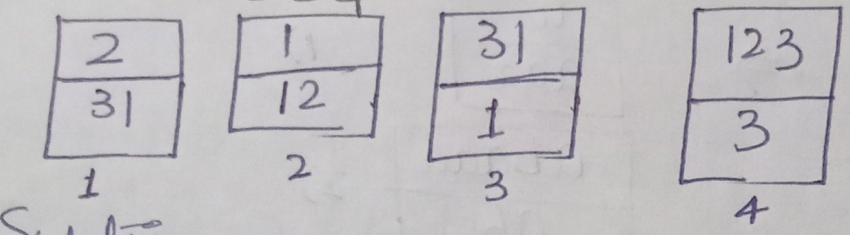




"A universal turing machine is a specified turing machine that can simulate the behaviour of any turing machine."

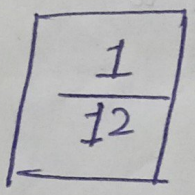
PCP Examples:

1.

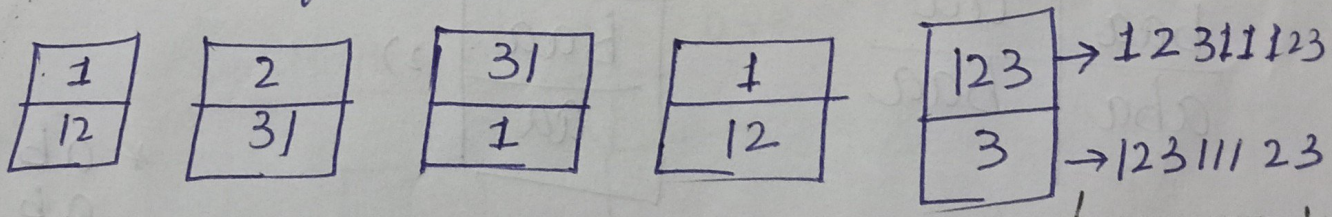


Solution: We need to arrange these given dominos in such a finite sequence that top part and bottom part contains the same pattern

We will select dominos '2' first, coz 'Value 1' is common in top and bottom, other dominos dont have common values.



⇒ In this '1' is common in top and bottom, now select a dominos which has 2 in ~~bottom~~ upper part, dominos 1 is the perfect choice



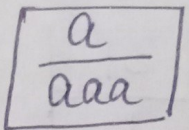
This problem is decidable

2)

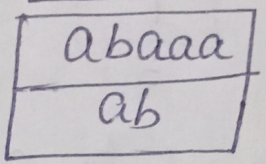
X	Y
a	aaa
abaaa	ab
ab	b

⇒

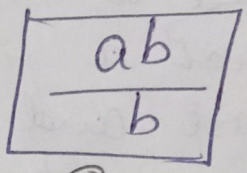
Domino



1)



2)



3)

	②	①	②	③
X:	ab	a	a	ab → abaaaaaab
Y:	ab	aaa	aaa	b → abaaaaaab

⇓

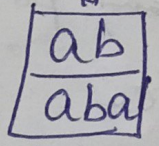
"Problem is decidable"

3)

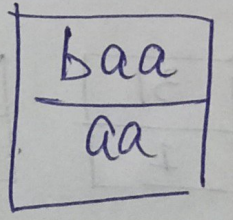
X	Y
ab	aba
baa	aa
aba	baa

Dominos

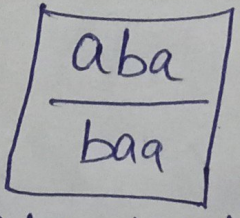
⇓



1)



2)



3)

* ab ab
  ab aa

	①	①x	③	③	.....	∞	undecidable
X:	ab	ab	aba	aba	.....	∞	
Y:	aba	aba	baa	baa	.....	∞	

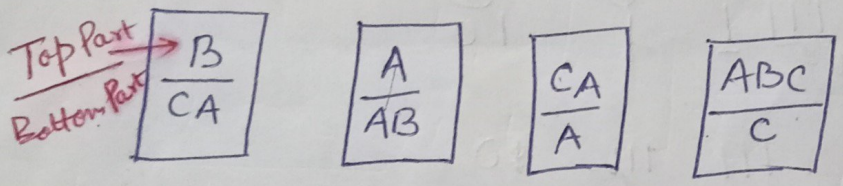
⇓

This domino can not be selected coz no match to 'b'

Post Correspondence Problem:

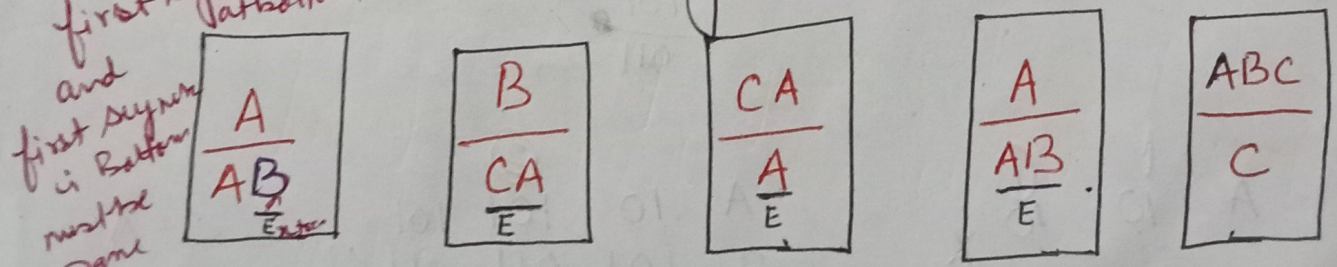
The post correspondence problem is an undecidable decision that was introduced by Emil Post in 1946.

Dominos: / Tiles:



* Top part / Bottom Part *

Task: We need to find a finite sequence of dominos such that the top and bottom strings are the same.



* It must be finite sequence

→ A B C A A A B C

→ A B C A A A B C

We Solved this problem.

E-2

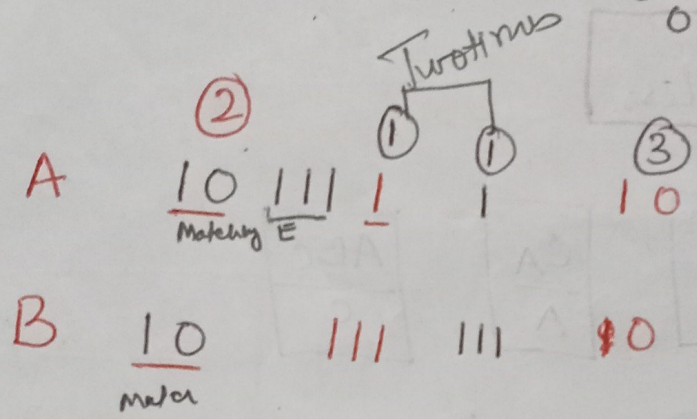
	A	B
①	1	111
②	10 111	10
③	10	0

$\Rightarrow$

$\frac{1}{111}$

$\frac{10\ 111}{10}$

$\frac{10}{0}$



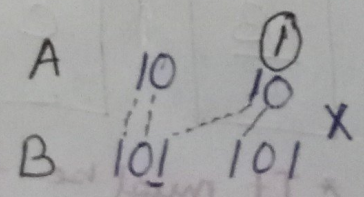
E-3

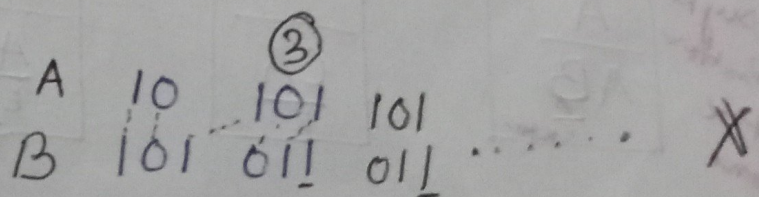
	A	B
①	10	101
②	011	11
③	101	011

$\frac{10}{101}$ ✓

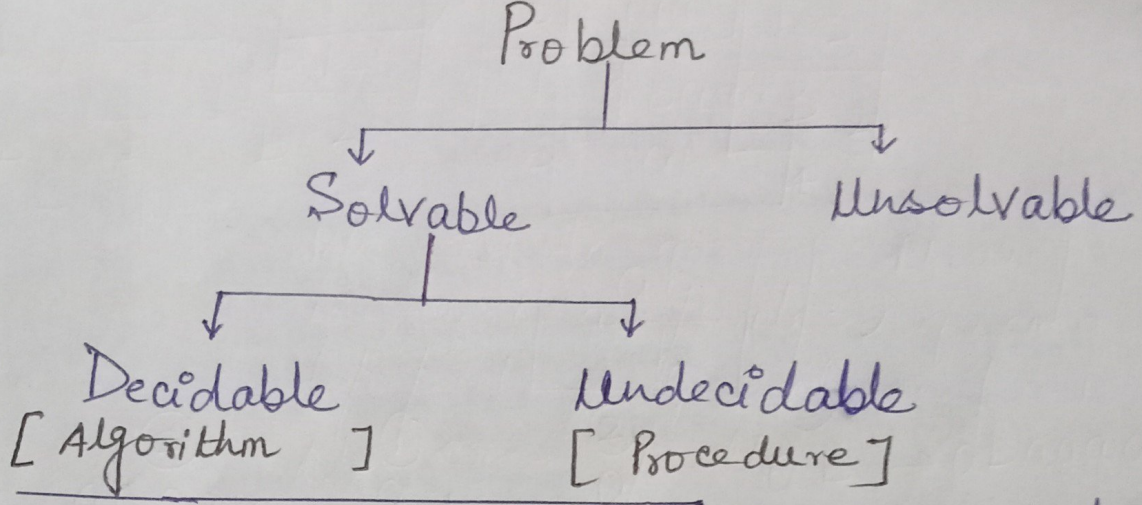
$\frac{011}{11}$

$\frac{101}{011}$





Decidable and Undecidable:



**Unsolvable: Neither we can solve the problem nor we are in condition to say that problem can not be solved.

*Solvable: A problem is said to be solvable, if we are able to find a solution.
A problem is also solvable if we can prove mathematically that problem is not solvable

* Procedure ⇒ all steps are known but not bound to time means no time frame is mentioned

* Algorithm ⇒ all steps are known with some time frame.

Universal Turing Machine

Universal Turing machine can be describe using multi-tape Turing machine defined as

$M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q: Finite set of states

Σ : is a finite set of symbols called the input alphabet

Γ : is a finite set of symbols called the tape alphabet

δ : Transition Function defined as

$$Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

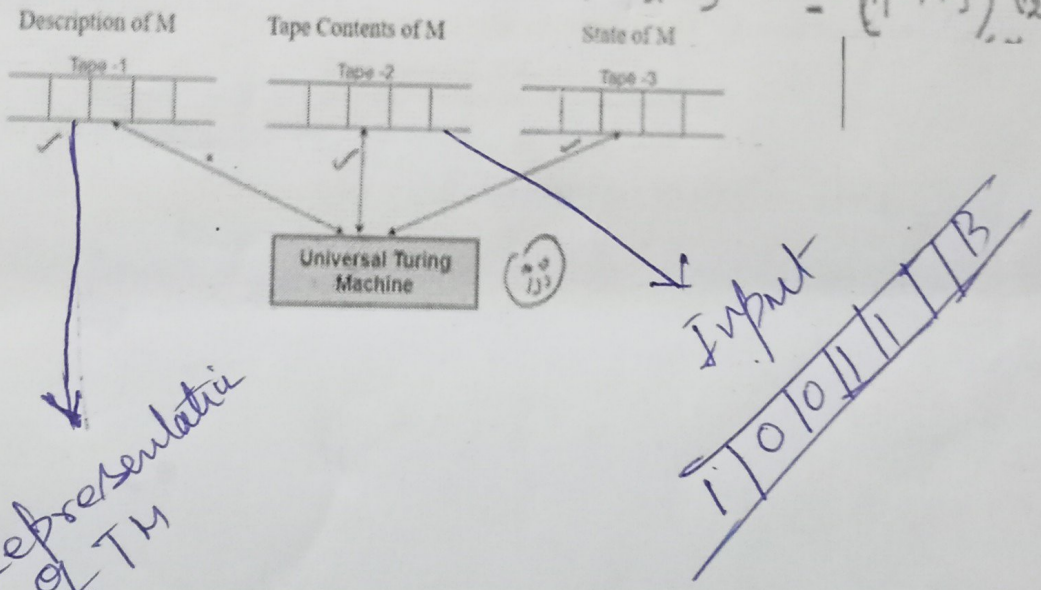
$$\delta : Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \rightarrow Q \times (\Gamma_1 \times \{L,R\}) \times (\Gamma_2 \times \{L,R\}) \times (\Gamma_3 \times \{L,R\})$$

q_0 : Initial state, $q_0 \in Q$

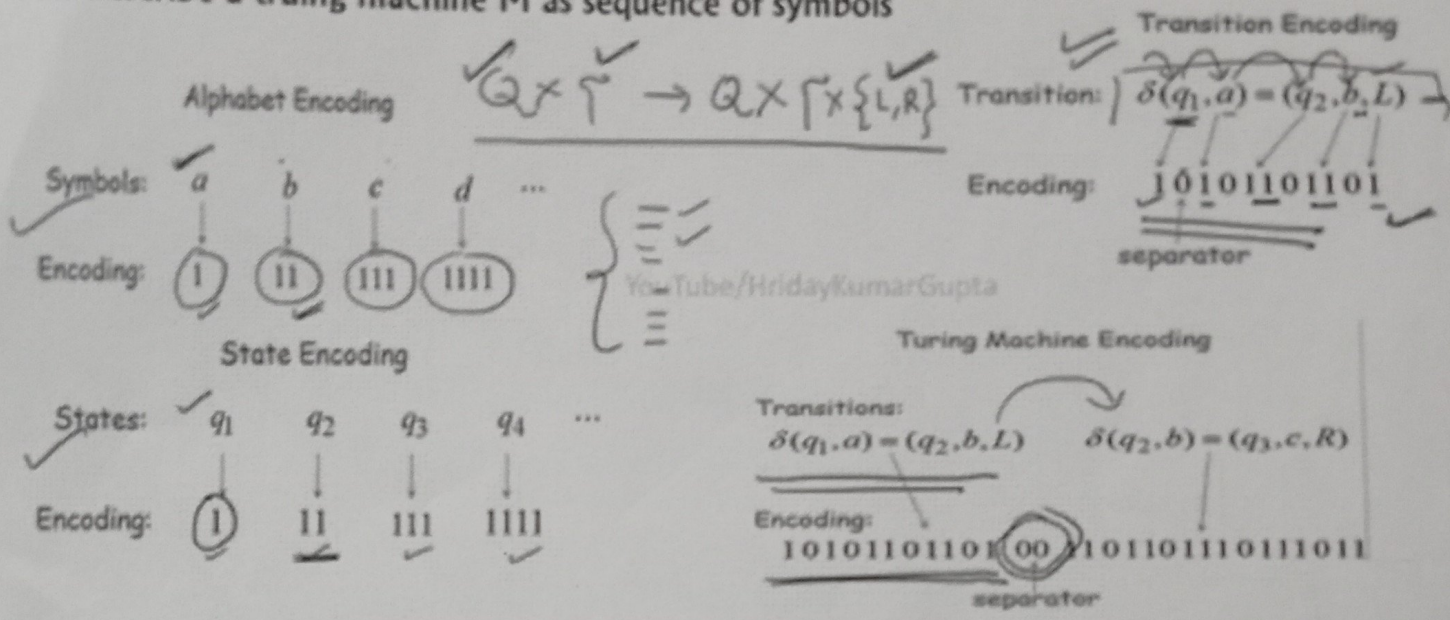
B : special symbol called the blank, $B \in \Gamma$

F : set of final state(s), $F \subseteq Q$

$$Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \rightarrow Q \times (\Gamma_1 \times \{L,R\}) \times (\Gamma_2 \times \{L,R\})$$



Lets describe a truing machine M as sequence of symbols



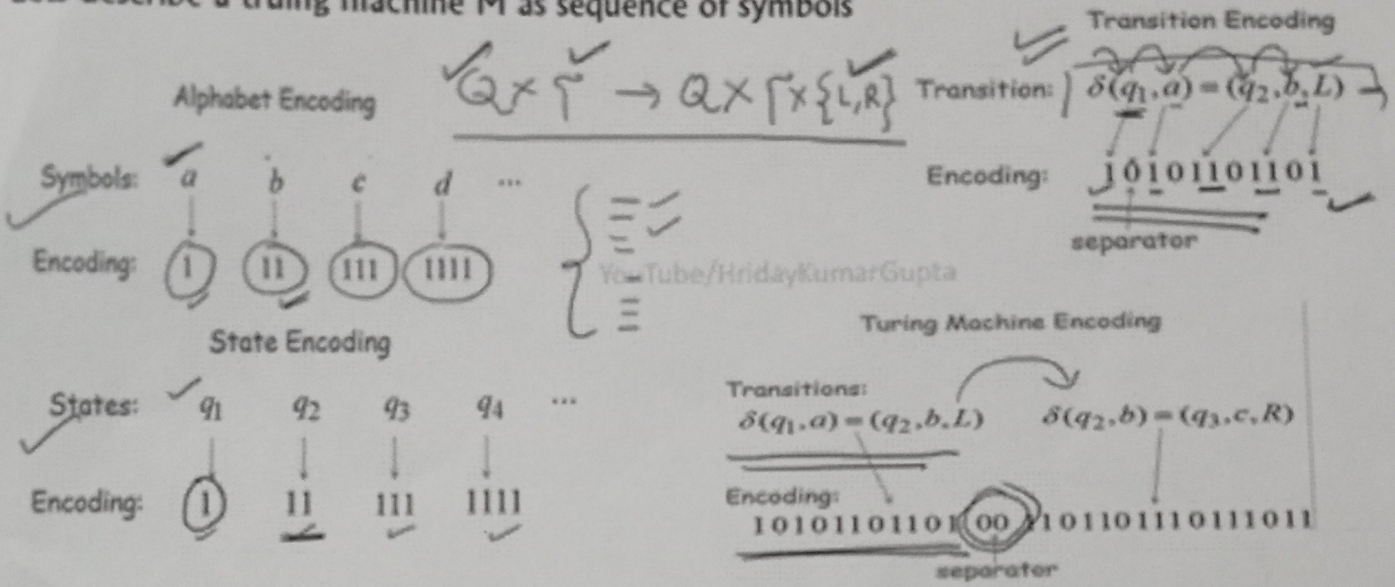
Tape 1 contents of Universal Turing Machine:

binary encoding
of the simulated machine M

Tape 1

1 0 1 0 11 0 11 0 10011 0 1 10 111 0 111 0 1100...

Lets describe a truing machine M as sequence of symbols



Decidability and Undecidability:

* Recursive Languages
* Recursively Enumerable Languages

Decidable Languages: A Language L is decidable if it is a "recursive language". All decidable languages are recursives and Vice-Versa.

Partially Decidable languages: A language L is said to be partially decidable if L is a recursively enumerable language.

Undecidable Languages:

1. A language is undecidable if it is not decidable
2. An undecidable language may sometimes

be partially decidable but not decidable

3. If a language is not even partially decidable, then there exist no TM for that language.

The Halting Problem:

*Algorithm

Given a program, WILL IT HALT?

"Given a TM, will it halt when run on some particular given input string"

*No algorithm

"Given some program written in some language (Java/C/etc) will it ever get into an infinite loop or will it always terminate"

Answer: 1. In general we can't always know

2. The best we can do is run the program and see whether it halt

"But for program in general the question is undecidable"