Unit IV - Turing Machine

Introduction:

Alan Turing is father of such a model which has computing capability of general purpose Computer.

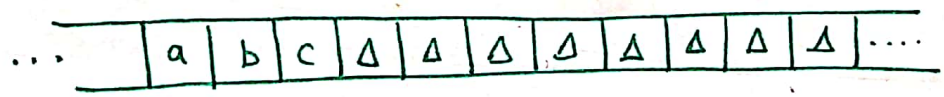
Turing Machine has following features:

1. It has External memory
2. It has unlimited memory capability
3. Input at left or right on the tape can be read easily.
4. The machine can produce certain output based on its input.

Model of Turing Machine:

The turing machine can be modelled with the help of following representation.

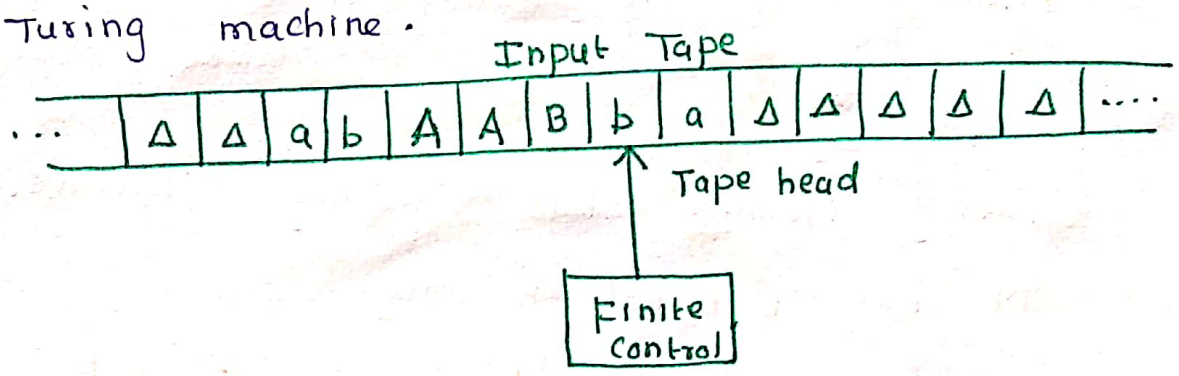
1) The input tape having infinite number of cells, each cell containing one input symbol, and thus the input string can be placed on a tape. The empty tape is filled by blank characters.



Input Tape

2) The finite control and the tape head which is responsible for reading the current input symbol. The tape head can move to left or right.
3. A finite set of states through which machine has to undergo.

4) Finite set of Symbols called external Symbols, which are used in building the logic of Turing machine.



Definition of Turing Machine (TM)

The turing machine is a collection of following components.

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \Delta \text{ or } B, F)$$

Q - finite set of states.

Γ - finite set of external Symbols.

Σ - finite set of input Symbols.

Δ (or) b (or) B - blank Symbol

δ - Transition or mapping function.

q_0 - Initial States

F - Finite set of States

General form of mapping function

$$(q_0, a) \rightarrow (q_1, A, L)$$

It is currently reading the input Symbol 'a' in the State 'q_0' then it goes to 'q_1' state by replacing (or) printing 'a' by 'A' and move ahead to left.

Techniques for Turing Machine Construction:

Construction of turing machine is a process of writing out the complete set of states and next move function. TM can be designed with the help of tools.

1. Storage in Finite Control
2. Multiple Tracks
3. Checking of Symbols
4. Subroutine.

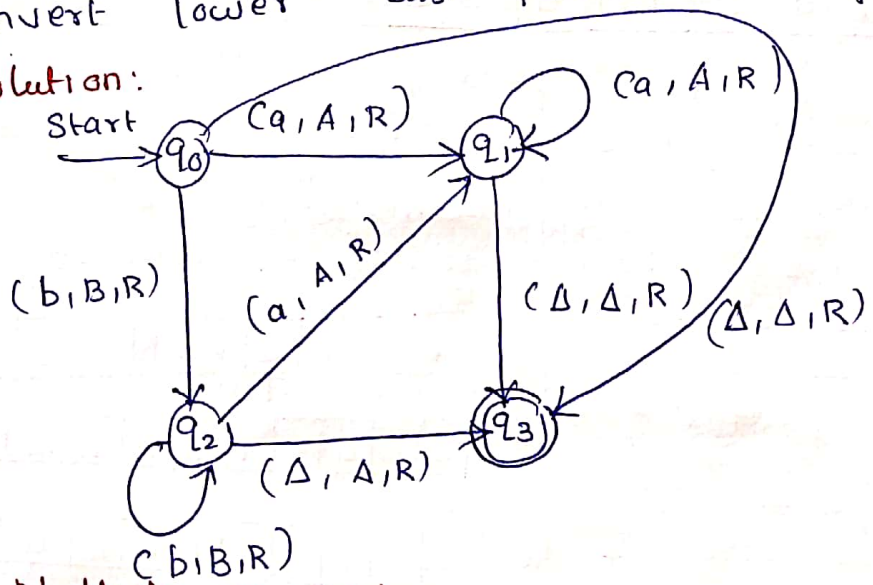
(i) Storage in Finite control:

The model of TM has a finite control. This finite control can be used to store (or) hold some amount of data (or) information.

Example:

Construct a TM M for $\Sigma = \{a, b\}$ which will convert lower case letters to upper case.

Solution:



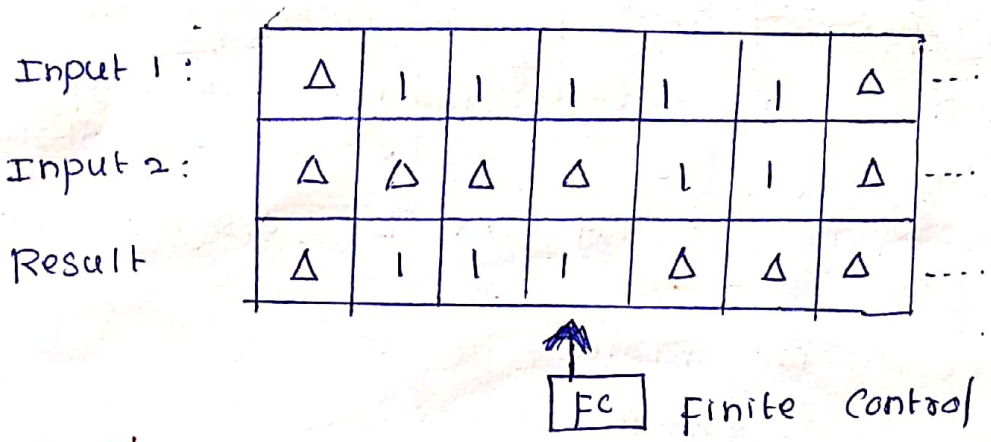
(ii) Multiple Tracks:

Input tape is divided into multiple tracks.

Example:

Subtractions of two unary numbers.

Let: $5 - 2 = 3 \Rightarrow 11111 - 11 = 111$



(iii) Checking off Symbols:

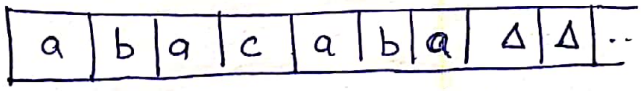
Checking of symbols is an effective way of recognizing the language by TM.

Example:

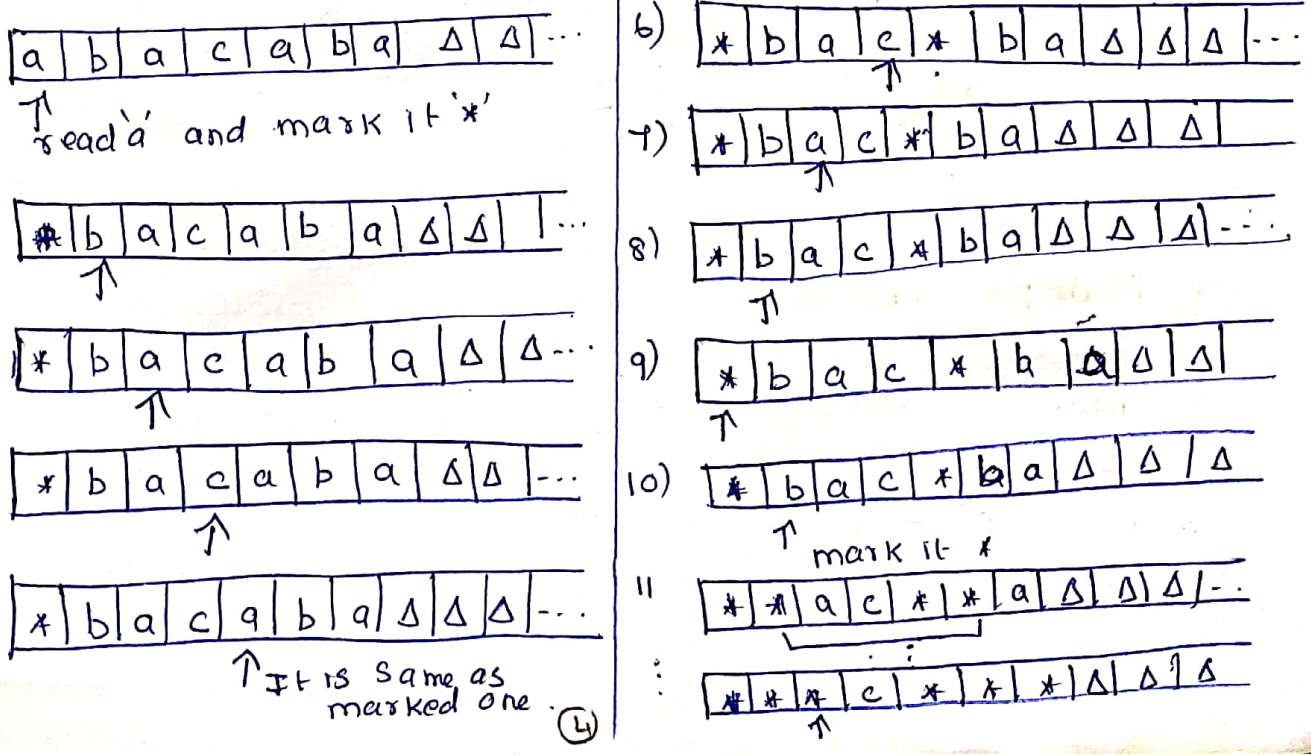
Construct a turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ which recognizes the language $L = \{wcw \mid w \in (a+b)^+\}$

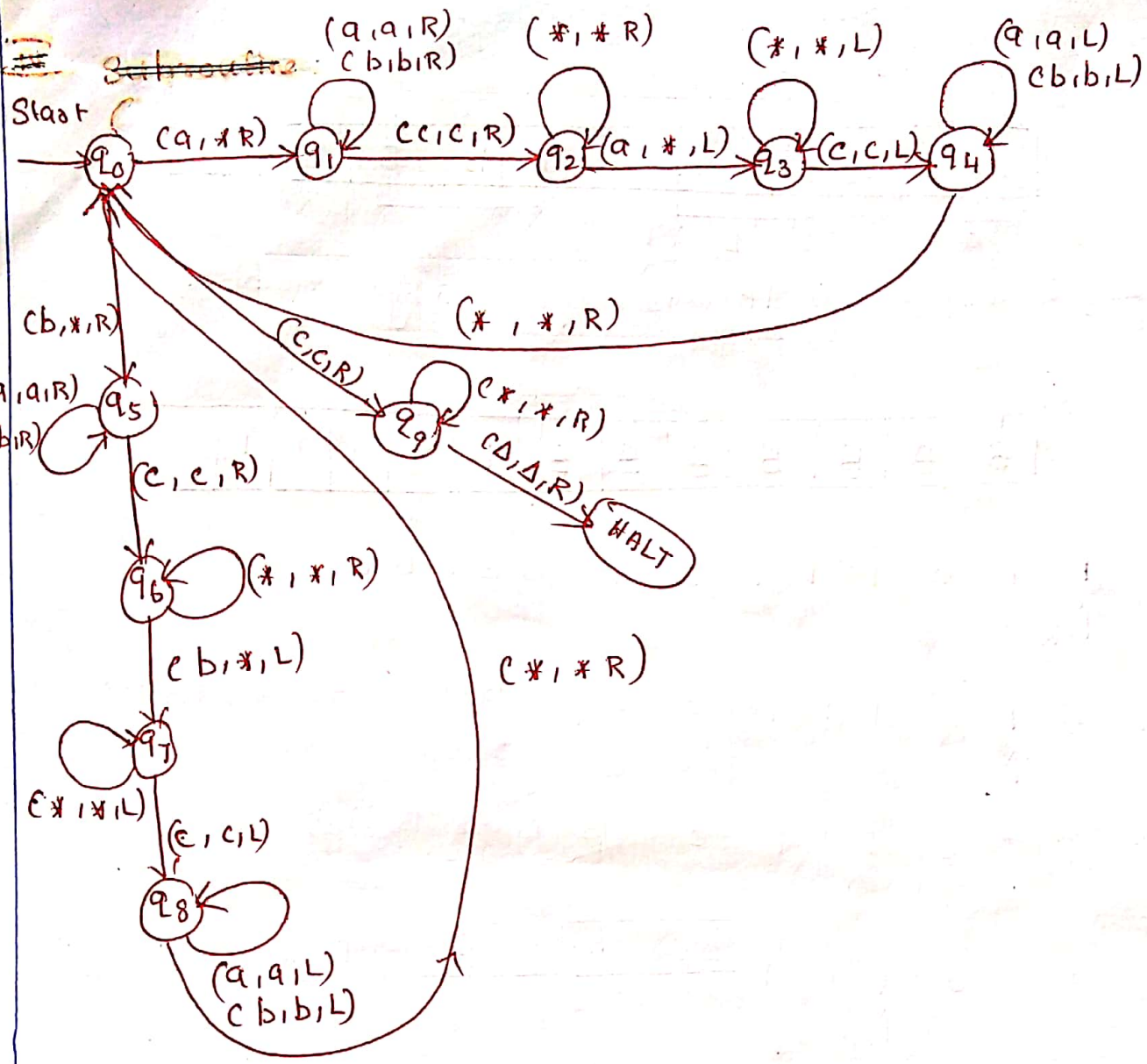
Solution:

$L = \{c, aca, bcb, abcab, bacba, abacaba \ldots\}$



Logic: Mark the first letter and then move to right till not get 'c', the first letter after 'c' will be compared with the marked letter. If it is same then mark this symbol, otherwise, go to reject state.





IV Subroutine:

In the high level languages use of subroutines built the modularity in the program development process. The same type of concept can be introduced in construction of TM.

X. partB. (1)

Example:

Construct a TM for the subroutine

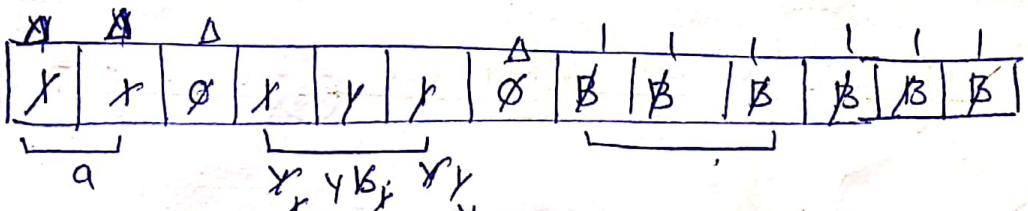
$f(a,b) = a * b$.

where a and b are unary numbers.

Solution:

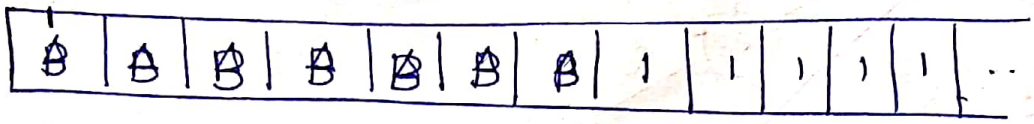
Let $a = 2$, $b = 3$ then.

$11 \times 111 \Rightarrow 111111$



Logic: copy 'b' inputs after 'a' by 'a' number of times.

output:



1) B
   X 1 0 1 1 1 0 B B B B B B
   ↑R

2) B 1 0 1 1 1 0 B B B B B B
     ↑R

3) B 1 0 1 1 1 0 B B B B B B
       ↑R

4) B 1 0 Y 1 1 0 B B B B B B
         ↑R

5) B 1 0 Y 1 1 0 B B B B B B
           ↑R

6) B 1 0 Y 1 1 0 B B B B B B
             ↑R

7) B 1 0 Y 1 1 0 B B B B B B
               ↑R

8) B 1 0 Y 1 1 0 B B B B B B
             ↑L

9) B 1 0 Y 1 1 0 1 B B B B B
           ↑L

10) B 1 0 Y 1 1 0 1 B B B B B
         ↑L

11) B 1 0 Y 1 1 0 1 B B B B B
       ↑L

12) B 1 0 Y 1 1 0 1 B B B B B
↑R

13) B 1 0 Y Y 1 0 1 B B B B B
↑R

14) B 1 0 Y Y 1 0 1 B B B B B
↑R

15) B 1 0 Y Y 1 0 1 B B B B B
↑R

16) B 1 0 Y Y 1 0 1 B B B B B
↑R

17) B 1 0 Y Y 1 0 1 B B B B B
↑L

18) B 1 0 Y Y 1 0 1 1 B B B B

19) B 1 0 Y Y 1 0 1 1 B B B B
↑L

20) B 1 0 Y Y 1 0 1 1 B B B B
↑L

21) B 1 0 Y Y 1 0 1 1 B B B B
↑L

22) B 1 0 Y Y 1 0 1 1 B B B B
↑R

23) B 1 0 Y Y Y 0 1 1 B B B B
↑R

24) B 1 0 Y Y Y 0 1 1 B B B B
↑R

25) B 1 0 Y Y Y 0 1 1 B B B B
↑R

26) B 1 0 Y Y Y 0 1 1 B B B B
↑R

27) B 1 0 Y Y Y 0 1 1 B B B B
↑L

28) B 1 0 Y Y Y 0 1 1 1 B B B
↑L

29) B 1 0 Y Y Y 0 1 1 1 B B B
↑L

30) B 1 0 Y Y Y 0 1 1 1 B B B
↑L

31 B 1 0 Y Y Y 0 1 1 1 B B B
↑L

32) B 1 0 Y Y 1 0 1 1 1 B B B
↑L

33) B 1 0 Y Y 1 0 1 1 1 B B B
↑

34) B 1 0 Y 1 1 0 1 1 1 B B B
↑L

35) B 1 0 1 1 1 0 1 1 1 B B B
↑L

36) B 1 0 1 1 1 0 1 1 1 B B B
↑L

37) B 1 0 1 1 1 0 1 1 1 B B B
↑R

38) Repeat the Step. 1 to 28.

39) B B 0 1 1 1 0 1 1 1 1 1 1
↑R

40) B B Ø B 1 1 1 0 1 1 1 1 1 1
↑R B

41 B B B X 1 1 0 1 1 1 1 1 1
↑R B

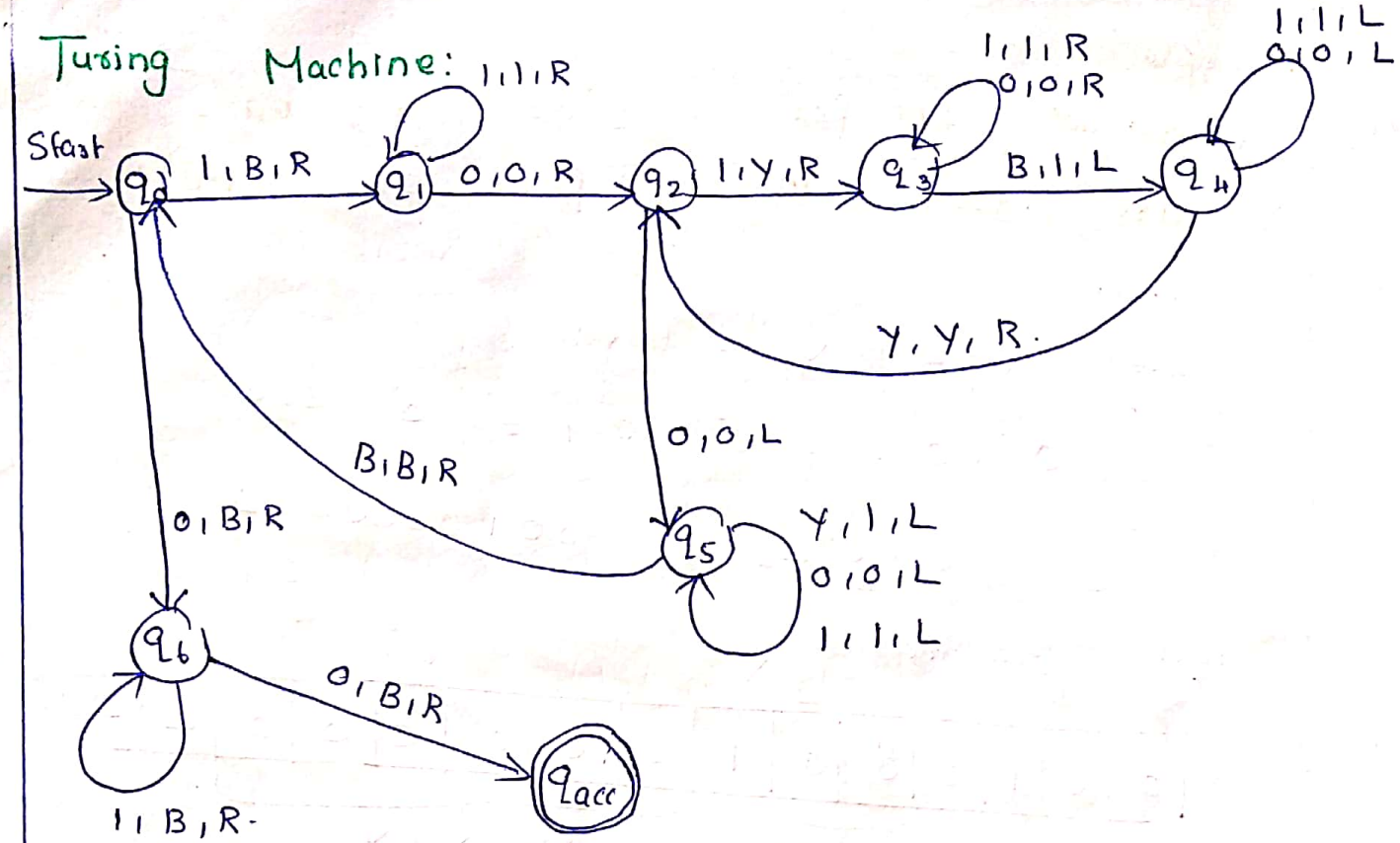
42) B B B B X 1 0 1 1 1 1 1 1
↑R B

43) B B B B B X 0 1 1 1 1 1 1
↑R B

44) B B B B B B Ø 1 1 1 1 1 1
↑R

45) B B B B B B B 1 1 1 1 1 1
↑ Halt.

Turing Machine:



Transition Table:

	δ	0	1	Y	B
→	q_0	q_6 B R	q_1 B R	–	–
	q_1	q_2 0 R	q_1 1 R	–	–
	q_2	q_5 0 L	q_3 Y R	–	–
	q_3	q_3 0 R	q_3 1 R	–	q_4 1 L
	q_4	q_4 0 L	q_4 1 L	q_2 Y R	–
	q_5	q_5 0 L	q_5 1 L	q_5 1 L	q_0 B R
	q_6	q_{acc} B R	q_6 B R	–	–
*	q_{acc}	–	–	–	–

Example:

Construct TM for $f(x,y) = x * y$ where x and y are stored in the form of $0^x 1 0^y 1$

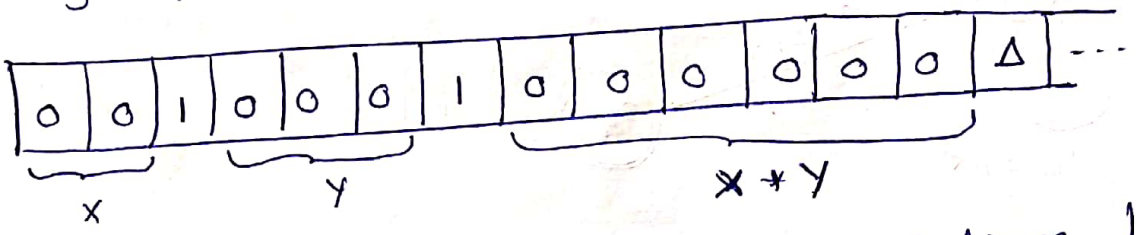
Solution:

Let $x = 2$, $y = 3$ then.

$$0^x 1 0^y 1 = 0^2 1 0^3 1 \Rightarrow 0010001$$

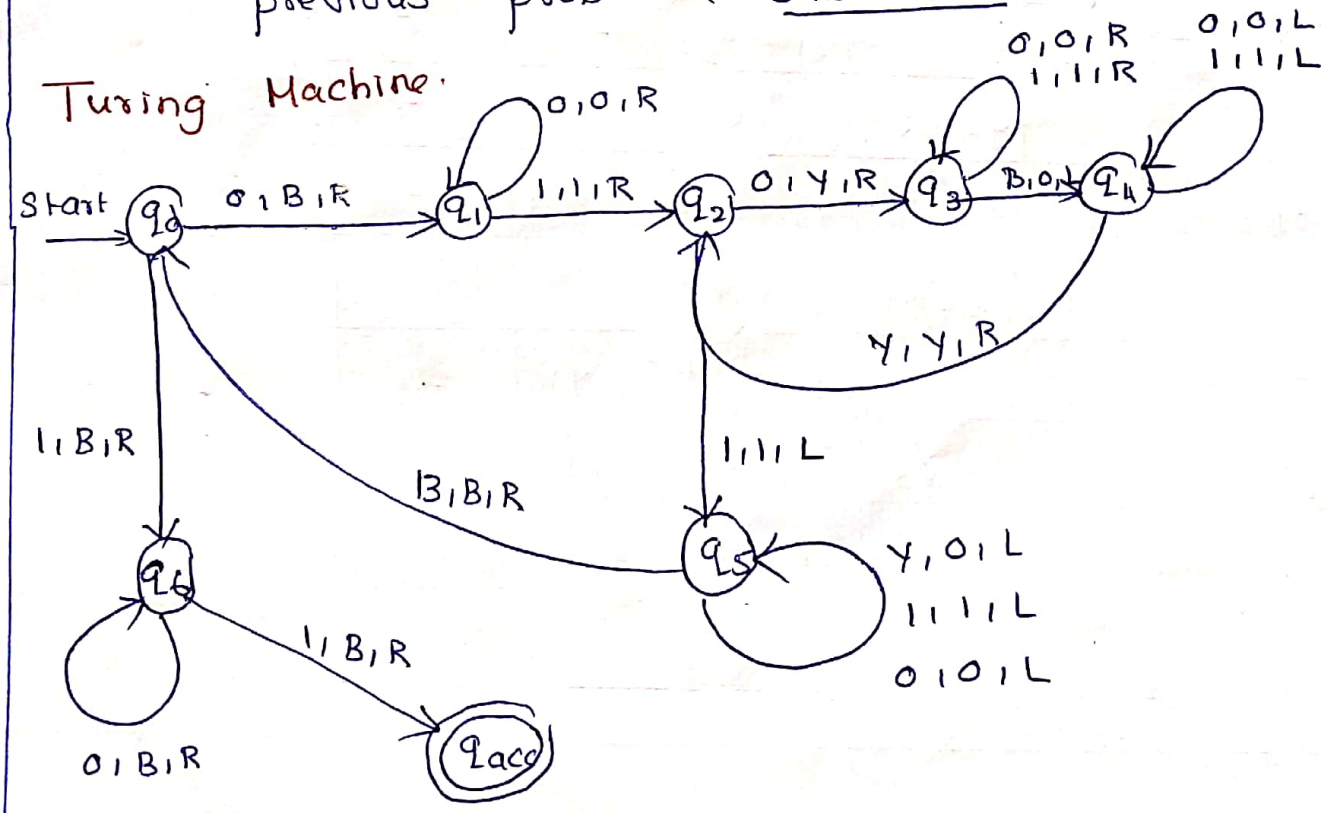
$\therefore f(x,y) = x * y \Rightarrow 00100010000000\Delta$.

Turing Machine



Logic: Copy y inputs by 'x' no. of times like previous problem (Subroutine).

Turing Machine.



Computable Languages and Functions.

The turing machine accepts all the languages even though there are recursively enumerable. Recursive means repeating the same set of rules for any number of times. Enumerable means a list of elements. The TM also accepts the computable functions, such as addition, multiplication, subtraction, division, power function, square function, logarithmic function and many more.

Example:

Construct a TM which accepts the language of aba over $\Sigma = \{a, b\}$.

Solution:

$$L = \{aba\}$$

We will assume the input tape the string 'aba' is:

The tape head read out the sequence upto the Δ character if 'aba' is readout the TM will halt after reading Δ.

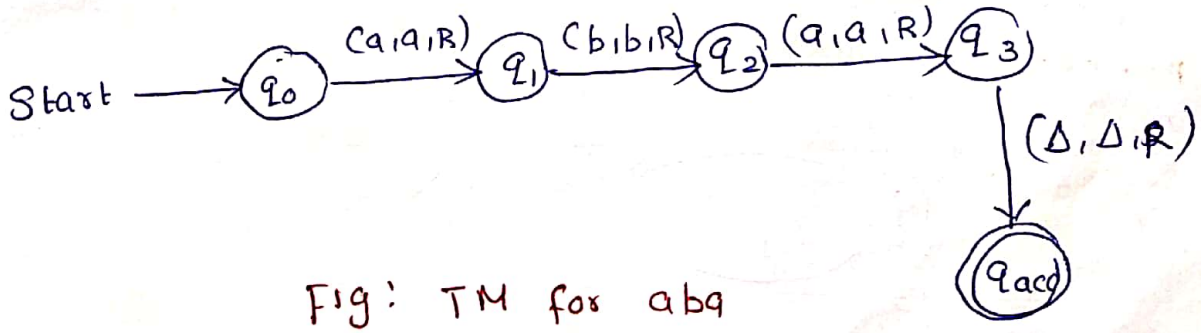


Fig: TM for aba

The same TM can be represented by another method of transition table.

	δ	a	b	Δ	
→	q_0	q_1 a, R	–	–	
	q_1	–	q_2, b, R	–	
	q_2	q_3 a, R	–	–	
	q_3	–	–	q_{acc} Δ R	
*	q_{acc}	–	–	–	

In the given transition table, we write the triplet in each row as:

Next state, output to be printed, direction.

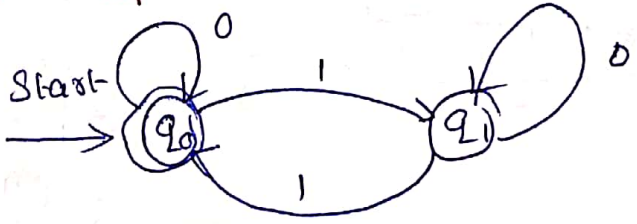
Thus TM can be represented by any of these methods.

Example:

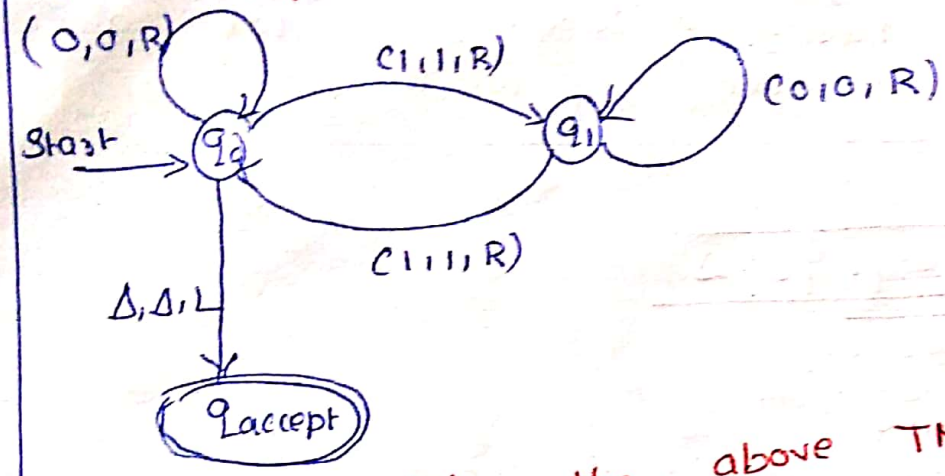
Construct TM for language consisting of strings having any number of 0's and only even number of 1's over the input set $\Sigma = \{0, 1\}$

Solution:

FSM for even number of ~~zeros~~ one's and any number of one's



Convert FSM into TM:



Let us Simulate the above TM for the input

1 1 0 1 0 1

1 1 0 1 0 1 Δ
↑ R
q_0

1 1 0 1 0 1 Δ
↑ R.

1 1 0 1 0 1 Δ
↑ R.

1 1 0 1 0 1 Δ
↑ R.

1 1 0 1 0 1 Δ
↑ R.

1 1 0 1 0 1 Δ
↑

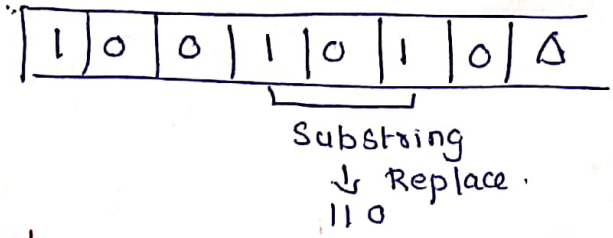
1 1 0 1 0 1 Δ
↑ Halt.

Thus the input String is accepted by TM.

Example:

Design a Turing machine which recognizes the input languages having a substring as 101 and replaces every occurence of 101 by 110.

Solution:



Let us simulate it for the input

1 0 0 1 0 1 0 Δ read 1 print 1 go to q_1
↑ R

1 0 0 1 0 1 0 Δ go to R and next state is q_2
  ↑ R

1 0 0 1 0 1 0 Δ
    ↑ R

1 0 0 1 0 1 0 Δ
      ↑ R

1 0 0 1 0 1 0 Δ Replace 0 → 1
        ↑ R

1 0 0 1 1 1 0 Δ
          ↑ R

1 0 0 1 1 1 0 Δ
            ↑ L

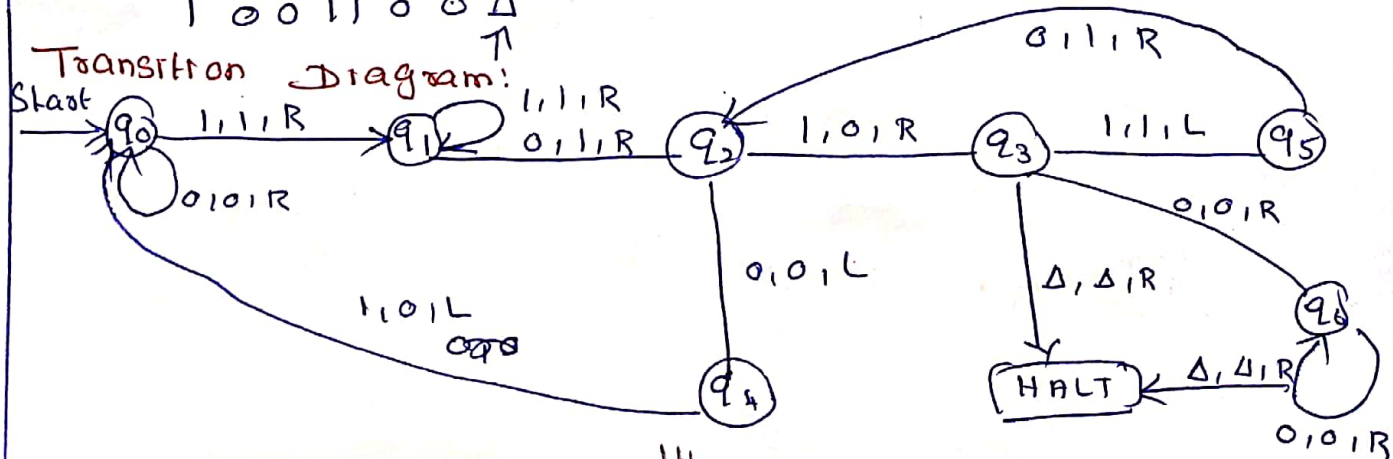
1 0 0 1 1 0 0 Δ
          ↑ R

1 0 0 1 1 0 0 Δ
            ↑

1 0 0 1 1 0 0 Δ
              ↑

Δ is a accepted State.

Transition Diagram:

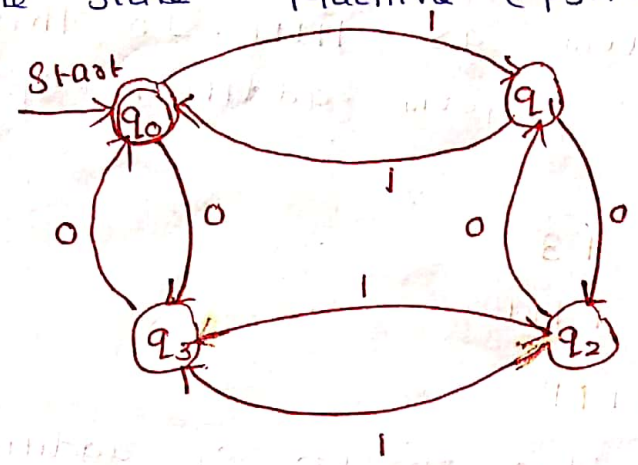


Part B
2

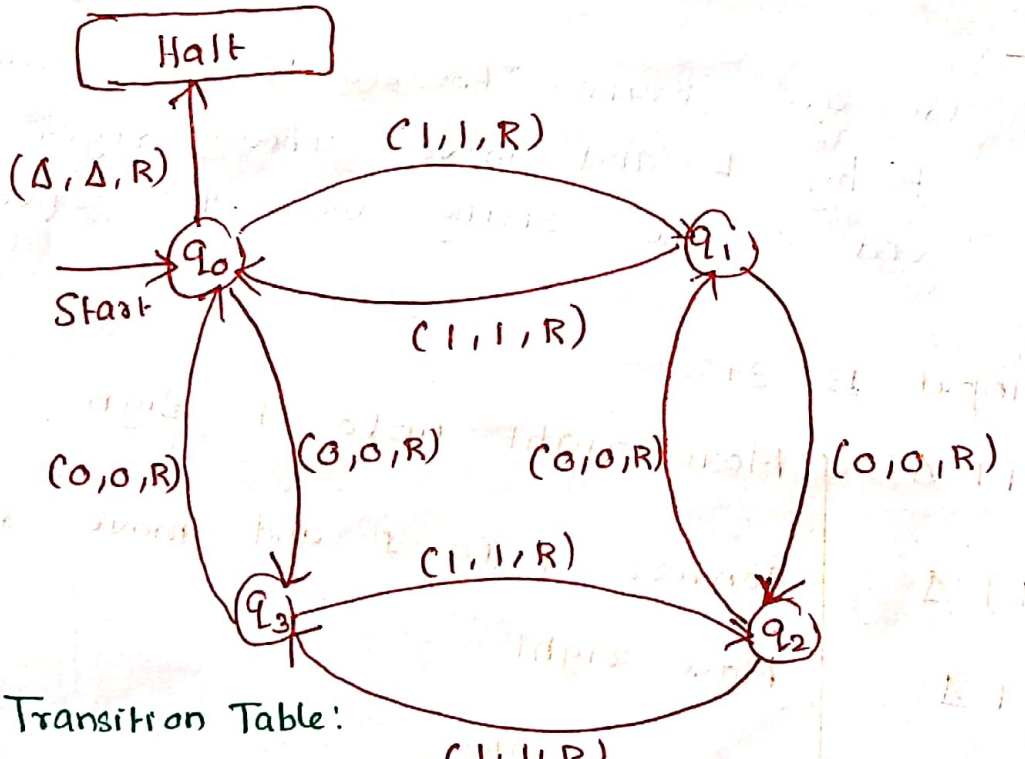
Construct a Turing machine for the language of even number of 1's and even number of 0's over $\Sigma = \{0,1\}$

Solution:

For this type of problem even we have drawn Finite State Machine (FSM).



And now it will be very much easy for us to convert this FSM to Turing Machine (TM).



Transition Table:

State	0	1	Δ
q_0	$q_3, 0, R$	$q_1, 1, R$	HALT, Δ, R
q_1	$q_2, 0, R$	$q_0, 1, R$	–
q_2	$q_1, 0, R$	$q_3, 1, R$	–
q_3	$q_0, 0, R$	$q_2, 1, R$	–
HALT	–	–	–

Part B.

3. Construct a TM for concatenation of the two strings of unary numbers.

Solution.

The unary number is made up of only one character i.e. The number 5 can be written in unary number system as 11111. In this TM we are going to perform addition of two unary numbers.

For example 2 + 3

i.e 11 + 111

= 11111

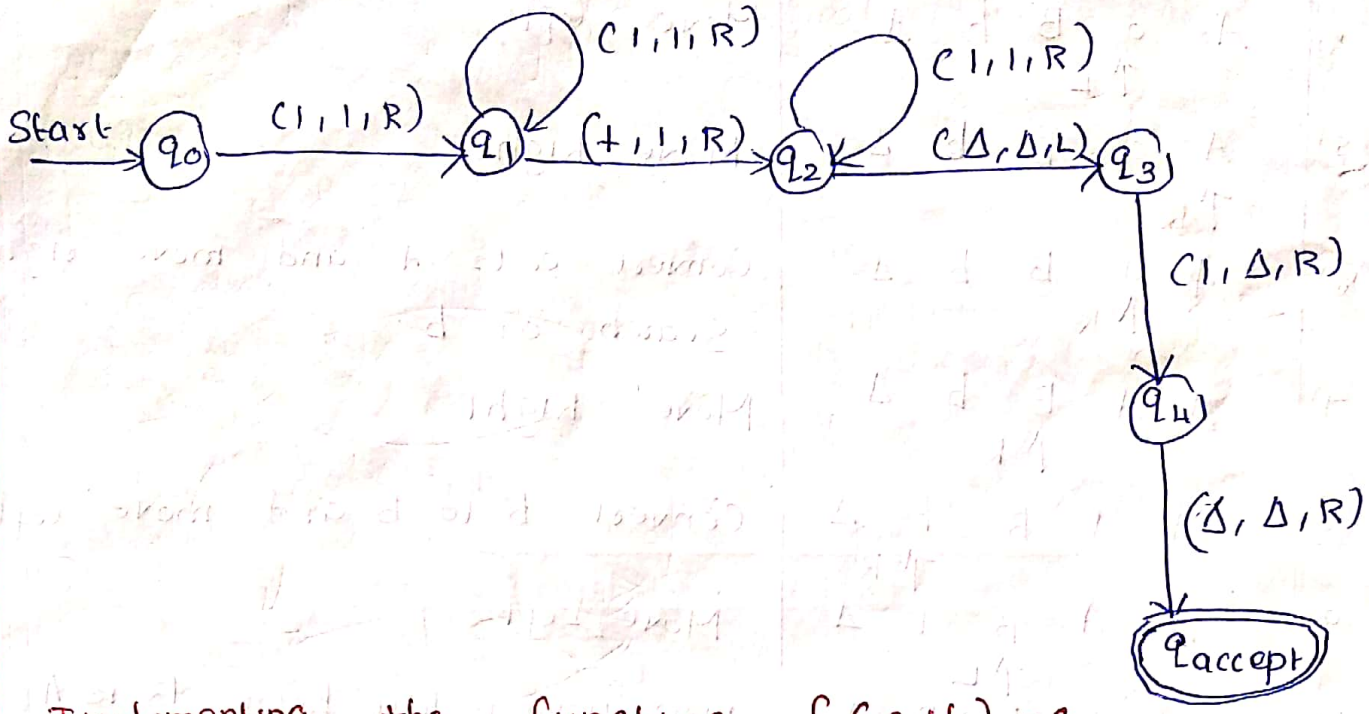
If you observe this process of addition, you will find the resemblance with string concatenation function.

What we are trying to do is, we simply replace + by 1 and move ahead right for Searching end of the string we will convert last 1 to Δ.

The input is 3+2

1 1 1 + 1 1 Δ ↑	Move right upto + sign
1 1 1 + 1 1 Δ ↑	Convert + to 1 and move right
1 1 1 1 1 1 Δ ↑	Move right
1 1 1 1 1 1 Δ ↑	Move right
1 1 1 1 1 1 Δ ↑	Now Δ has encountered so just move left
1 1 1 1 1 Δ ↑	Convert 1 to Δ
1 1 1 1 1 Δ Δ	Thus the tape now consist of the addition of two unary numbers.

The TM will look like this.



Implementing the function $f(a+b) = c$.
we assume a and b both are non zero elements

Transition Table.

	q	1	+	Δ
→	q_0	$q_1, 1, R$	–	–
	q_1	$q_1, 1, R$	$q_2, 1, R$	–
	q_2	$q_2, 1, R$	–	$q_3, Δ, L$
	q_3	q_4 Δ, R	–	–
	q_4	–	–	q_{acc} Δ, R.
*	q_{acc}	–	–	–

Part C

Example :

Construct TM for the language $L = \{a^n b^n\}$ where $n \geq 1$. $L = \{ab, aabb, aaabbb, \ldots\}$

Solution: The Simulation aabb can shown in below.

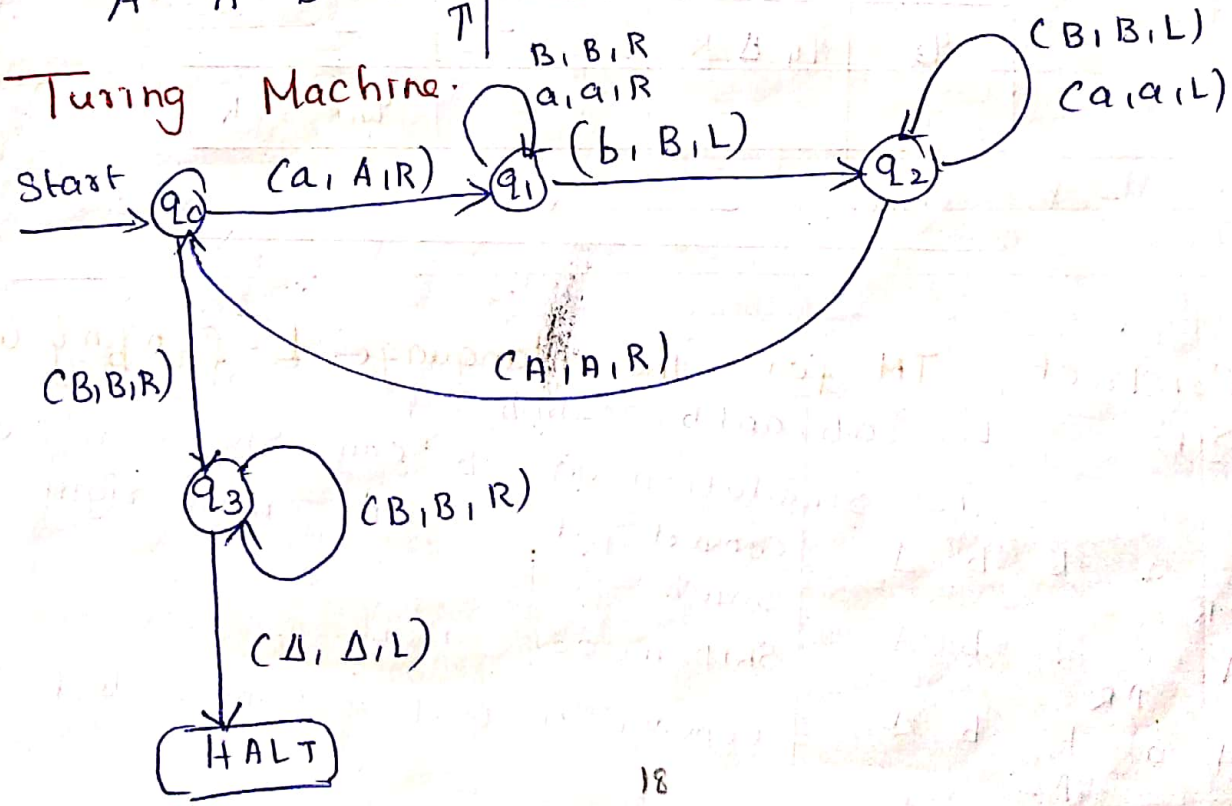
1) a a b b Δ
   ↑R
   Convert it A and move right in search of b.

2) A a b b Δ
     ↑R
   Skip it, move ahead

3) A a b b Δ
       ↑L
   Convert it to B and move left till A.

	~~A a R~~	
4)	A a B b Δ ↑L	Move left
5)	A a B b Δ ↑R	Move Right
6)	A a B b Δ ↑R	Convert a to A and move right in search of b.
7)	A A B b Δ ↑R	Move Right
8)	A A B b Δ ↑R	Convert b to B and move left.
9)	A A B B Δ ↑L	Move Left
10)	A A B B Δ ↑L	Now immediately before B is A that means all the a's are marked by A. So we will move right to ensure that no b is present.
11	A A B B Δ ↑R	Move Right
12.	A A B B Δ ↑R	Move Right
13	A A B B Δ ↑	Halt.

Turing Machine.



Transition Table:

→	δ	a	b	B	A	Δ
	q_0	q_1, A, R	—	q_3, B, R	—	—
	q_1	q_1, a, R	q_2, B, L	q_1, B, R	—	—
	q_2	q_2, a, L	—	q_2, B, L	q_0, A, R	—
	q_3	q_3, B, R	—	q_3, B, R		q_{acc}, Δ, L
*	q_{acc}	—	—	—	—	—

Construct a TM for $L = \{a^n b^n c^n \mid n \geq 1\}$

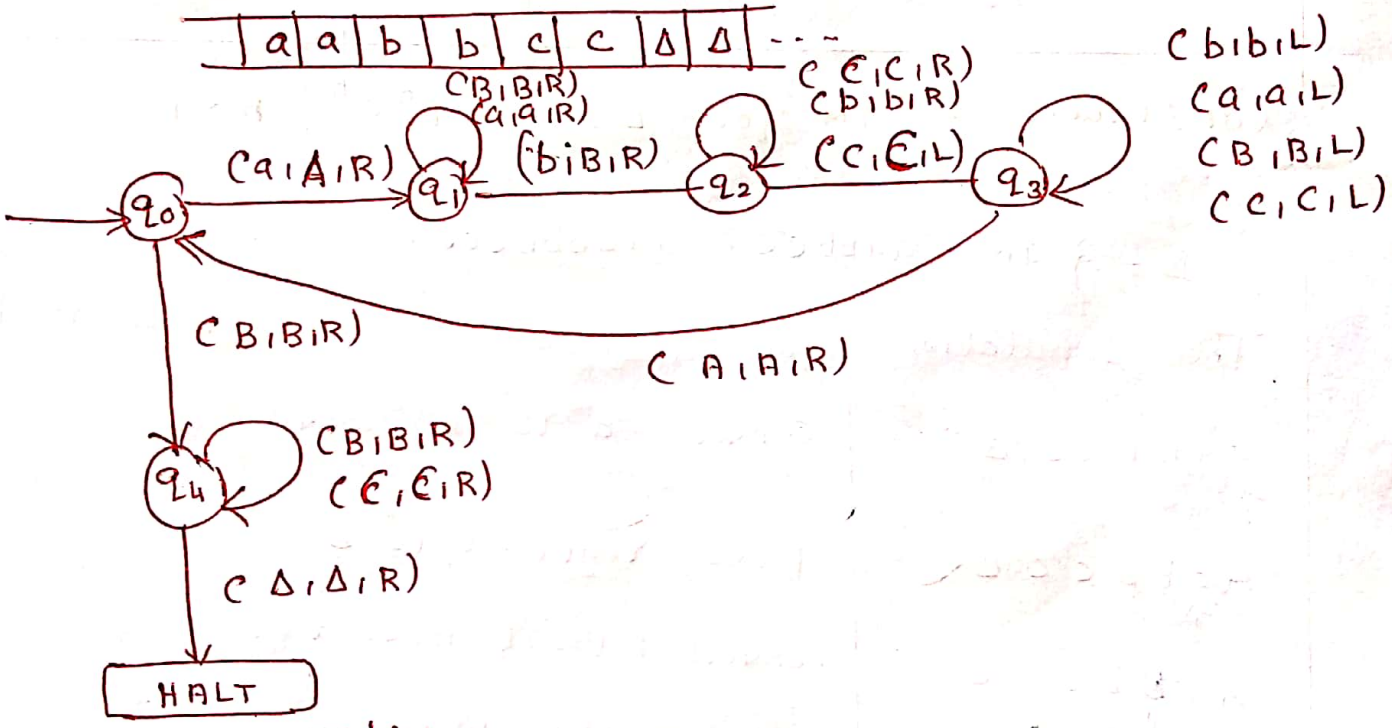
Solution:

$L = \{abc, aabbcc, aaabbbccc, \ldots\}$

The simulation for aabbcc can be shown as below.

a a b b c c Δ ↑	Convert a to A, move right
A a b b c c Δ ↑	Move right upto b
A a b b c c Δ ↑	Convert b to B move right
A a B b c c Δ ↑	move right upto c
A a B b c c Δ ↑	Convert c to C move right
A a B b C c Δ ↑	Move left upto A
A a B b C c Δ ↑	Move Right to a
A a B b C c Δ ↑	Convert a to A move right
A A B b C c Δ ↑	Move right
A A B b C c Δ ↑	Convert b to B move right

A A B B C C Δ ↑	Move right
A A B B C C Δ ↑	Convert C to C move ~~right~~ Left
A A B B C C Δ ↑	Move left up to A.
A A B B C C Δ ↑	Move Right Skip up to B,C
A A B B C C Δ ↑	Since Δ is read TM will reach to Halt state.



Transition Table:-

δ	a	A	b	B	C	C	Δ
q_0	q_1, A, R	–	–	q_4, B, R	–	–	–
q_1	q_1, a, R	–	q_2, B, R	q_1, B, R	–	–	–
q_2	q_2, C, R	–	q_2, b, R	–	q_3, C, L	q_2, C, R	–
q_3	q_3, a, L	q_0, A, R	q_3, b, L	q_3, B, L	–	q_3, C, L	–
q_4	–	–	–	q_4, B, R	–	q_4, C, R	HALT, Δ, R
HALT	–	–	–	–	–	–	–

Part B

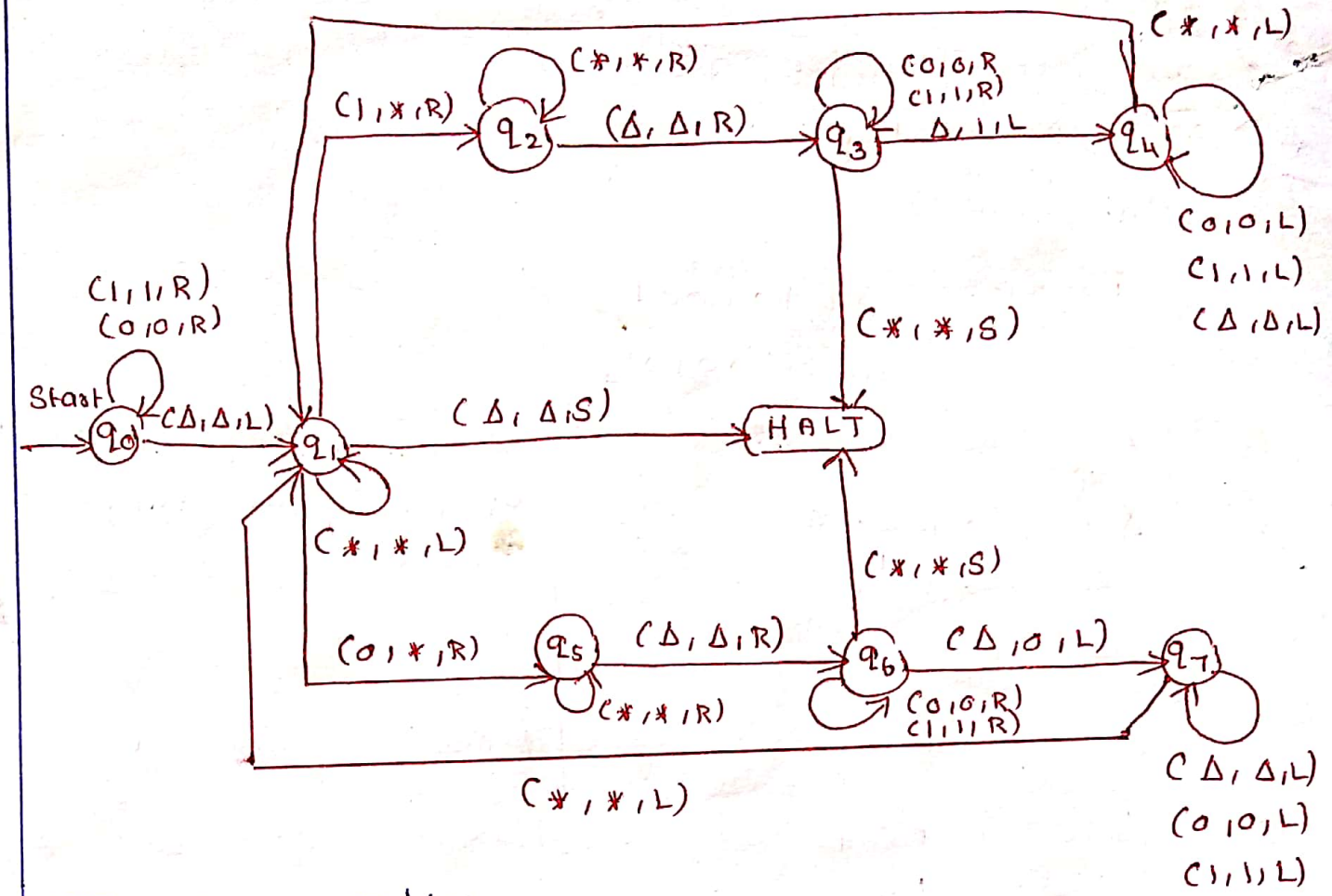
1) Construct TM for reversing a binary string on the input tape.

Solution:

Example $w = 101001$

$w^R = 100101$

... Δ 1 0 1 0 0 1 Δ Δ Δ	we will move upto the end of the input String.
Δ 1 0 1 0 0 1 Δ Δ Δ	move left
Δ 1 0 1 0 0 1 Δ Δ Δ	Mark it * and copy it after Δ.
Δ 1 0 1 0 0 * Δ 1	Move left upto *
Δ 1 0 1 0 0 * Δ 1 Δ	Move left covert 0 to * and place it at the end.
Δ 1 0 1 0 * * Δ 10 Δ	Move left upto *
Δ 1 0 1 0 * * Δ 10 Δ	Move left, convert 0 to * and place it at the end.
Δ 1 0 1 * * * Δ 100 Δ	Move left upto *
Δ 1 0 1 * * * Δ 100 Δ	Move left, convert 1 to * and place it at the end.
Δ 1 0 * * * * Δ 1001 Δ	Move left upto *
Δ 1 0 * * * * Δ 10010 Δ	convert it to *
Δ 1 * * * * * Δ 10010 Δ	and similarly copy 1 at the end of tape.
Δ * * * * * * Δ 100101 Δ	Move left skipping all *
Δ * * * * * * Δ 100101 Δ	Left to * is a Δ character so we goto HALT state.



Transition Table:

δ	0	1	*	Δ
q_0	$(q_0,0,R)$	$(q_0,1,R)$	–	(q_2,Δ,L)
q_1	$(q_5,*,R)$	$(q_2,*,R)$	$(q_1,*,L)$	HALT, Δ, S
q_2	–	–	$(q_2,*,R)$	(q_3,Δ,R)
q_3	$(q_3,0,R)$	$(q_3,1,R)$	–	$q_4,1,L$
q_4	$(q_4,0,L)$	$(q_4,1,L)$	$(q_1,*,L)$	q_4,Δ,L
q_5	–	–	$(q_5,*,R)$	$q_6,\Delta R$
q_6	$(q_6,0,R)$	$(q_6,1,R)$	–	$q_7,0,L$
q_7	$(q_7,0,L)$	$(q_7,0,L)$	$(q_1,*,L)$	q_7,Δ,L
HALT	–	–	–	–

Thus the TM reverses the input string.

Construct a TM for checking the palindrome of the String of even length.

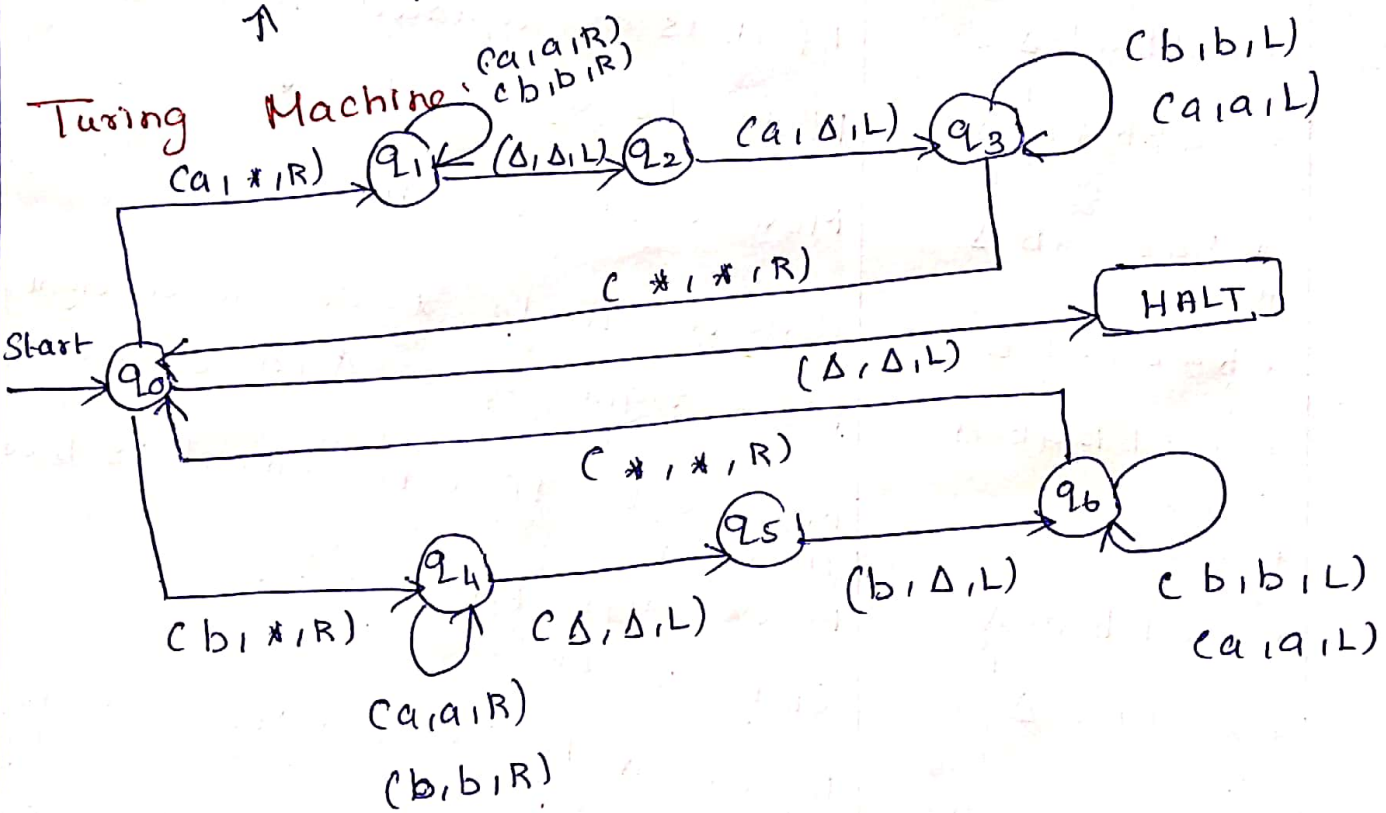
Solution:

The Simulation of machine for a valid input Such as:

a b a b b a b a Δ ↑	We will mark it by * and move to right end in search of a
* b a b b a b a Δ ↑	move right
* b a b b a b a Δ ↑	Moved right upto Δ
* b a b b a b a Δ ↑	Moved left, checked if it is a
* b a b b a b a Δ ↑	If it is a, replaced it by Δ
* b a b b a b Δ Δ ↑	Move to left, upto *
* b a b b a b Δ ↑	Move right, read it
* b a b b a b Δ ↑	Convert it to * and move right
* b a b b a b Δ ↑	Move right upto Δ in search of b
* * a b b a b Δ ↑	Move left, if left symbol is b convert it to Δ
* * a b b a b Δ ↑	Move left till *
* * a b b a Δ ↑	Goto right
* * a b b a Δ ↑	Replace a by * and move right upto Δ
* * a b b a Δ ↑	We will move left and check if it is a, then replace it by Δ.
* * * b b a Δ ↑	It is a so replace by Δ.
* * * b b a Δ ↑	Move to left till *
* * * b b Δ ↑	Move right.
* * * b b Δ ↑	

* * * b b Δ ↑ * * * * b Δ ↑ * * * * b Δ ↑ * * * * Δ ↑ * * * * Δ ↑ * * * * Δ ↑	Replace it by * Move right upto Δ Move left to see if it is b, if so then replace it by Δ. Move left till * Move right and check whether it is Δ, if so Goto HALT state.

Turing Machine:



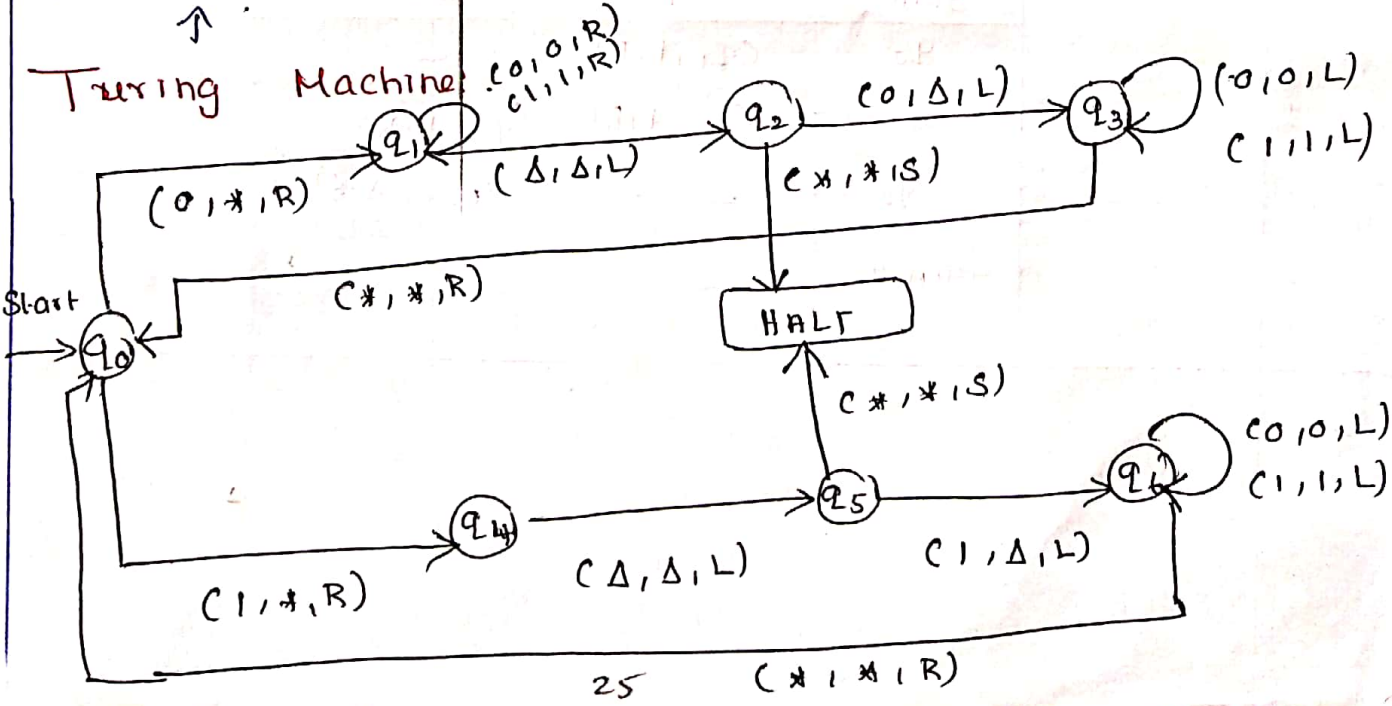
Construct a TM for checking the palindrome of a string of odd palindrome for $\Sigma = \{0,1\}$

Solution:

The simulation of machine for a valid input such as.

0 0 1 0 0 Δ ↑	Move right by making 0 to *
* 0 1 0 0 Δ ↑	Move right upto Δ
* 0 1 0 0 Δ ↑	Move left, replace 0 by Δ
* 0 1 0 Δ Δ ↑	Move left
* 0 1 0 Δ ↑	Move left upto *
* 0 1 0 Δ ↑	Move right convert 0 to * and then move right
* * 1 0 Δ ↑	Move right upto Δ
* * 1 0 Δ ↑	Move left and replace 0 by Δ
* * 1 Δ Δ ↑	Move left till *
* * 1 Δ Δ ↑	Move right and convert 1 to *
* * * Δ Δ ↑	Move right
* * * Δ Δ ↑	Move right
* * * Δ Δ ↑	Since it is * go to HALT State.

Turing Machine:



Construct a TM for a successor function for a given unary number i.e $f(n) = n+1$

Solution:

The input tape consists of 4 the successor function will give output as 5.

1 1 1 1 Δ Move to extreme right by keeping all 1s as it is

1 1 1 1 Δ Move right

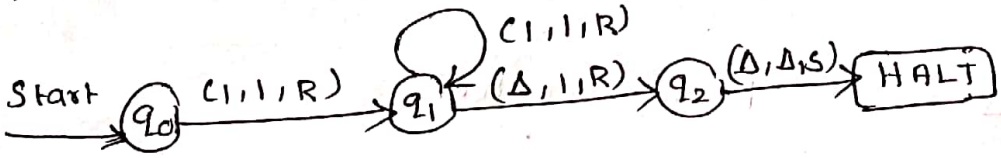
1 1 1 1 Δ Move right

1 1 1 1 Δ Move right

1 1 1 1 Δ Convert Δ to 1

1 1 1 1 1 Δ Halt

The construction of TM will be.



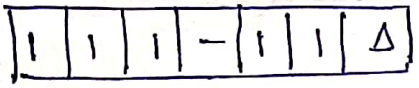
Transition Table:

States	1	Δ
q_0	$(q_1, 1, R)$	–
q_1	$(q_1, 1, R)$	$(q_2, 1, R)$
q_2	–	(HALT, Δ, S)
HALT	–	–

Construct TM for performing of two unary numbers $f(a-b) = c$ where a is always greater than b.

Solution:

Here we have certain assumption as first number is greater than second one.



Let $a = 3$, $b = 2$.

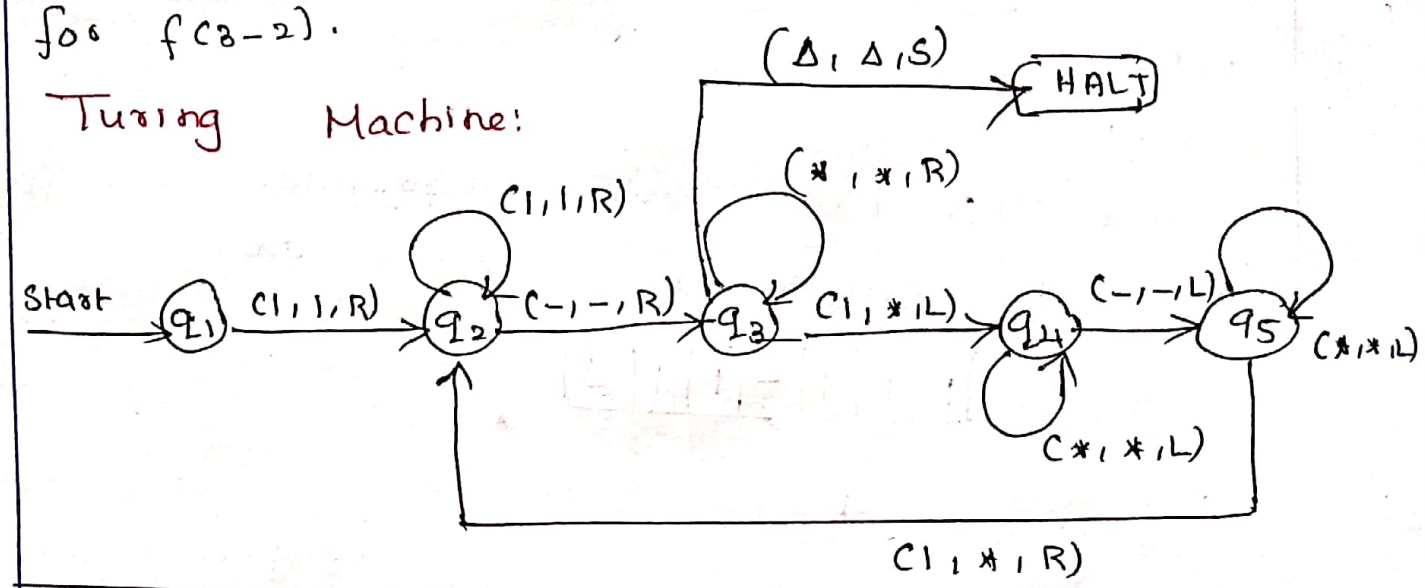
Move right to − symbol as perform reduction of number of 1's from first number

The simulation for understanding the logic.

111 − 11 Δ ↑	Move right upto '−'
111 − 11 Δ ↑	Move right and convert 1 to *
111 − * 1 Δ ↑	Now move left
111 − * 1 Δ ↑	Move left
11 1 − * 1 Δ ↑	Convert 1 to *
11 * − * 1 Δ ↑	Move right
11 * − * 1 Δ ↑	Move right till 1
11 * − * 1 Δ ↑	Convert 1 to * and move left
11 * − * * Δ ↑	Move left till 1
11 * − * * Δ ↑	Convert 1 to * and move right
1 * * − * * Δ ↑	Move right till Δ.
1 * − * * Δ ↑	HALT

Thus we get 1 on the input tape as answer for $f(3-2)$.

Turing Machine:



part B 2

Construct a TM for unary multiplication (Say 111 x 11 = 111111)

Solution:

$f(a,b) = a * b$

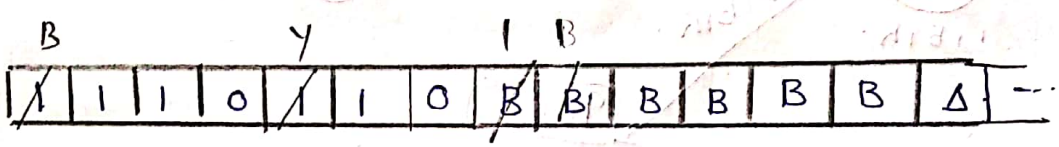
where a and b are unary numbers.

Let a = 3 b = 2 then.

111 x 11 => 111111

Logic:

Copy b inputs after 0 by 'a' number of times.



1) B
Y 1 1 0 1 1 0 B B B B B B Δ
↑ R

2) B 1 1 0 1 1 0 B B B B B B Δ
↑ R.

3) B 1 1 0 1 1 0 B B B B B B Δ
↑ R.

4) B 1 1 0 1 1 0 B B B B B B Δ
↑ R.

5) B 1 1 0 Y 1 0 B B B B B B Δ
↑ R

6) B 1 1 0 Y 1 0 B B B B B B Δ
↑ R

7) B 1 1 0 Y 1 0 B B B B B B Δ.
↑ R

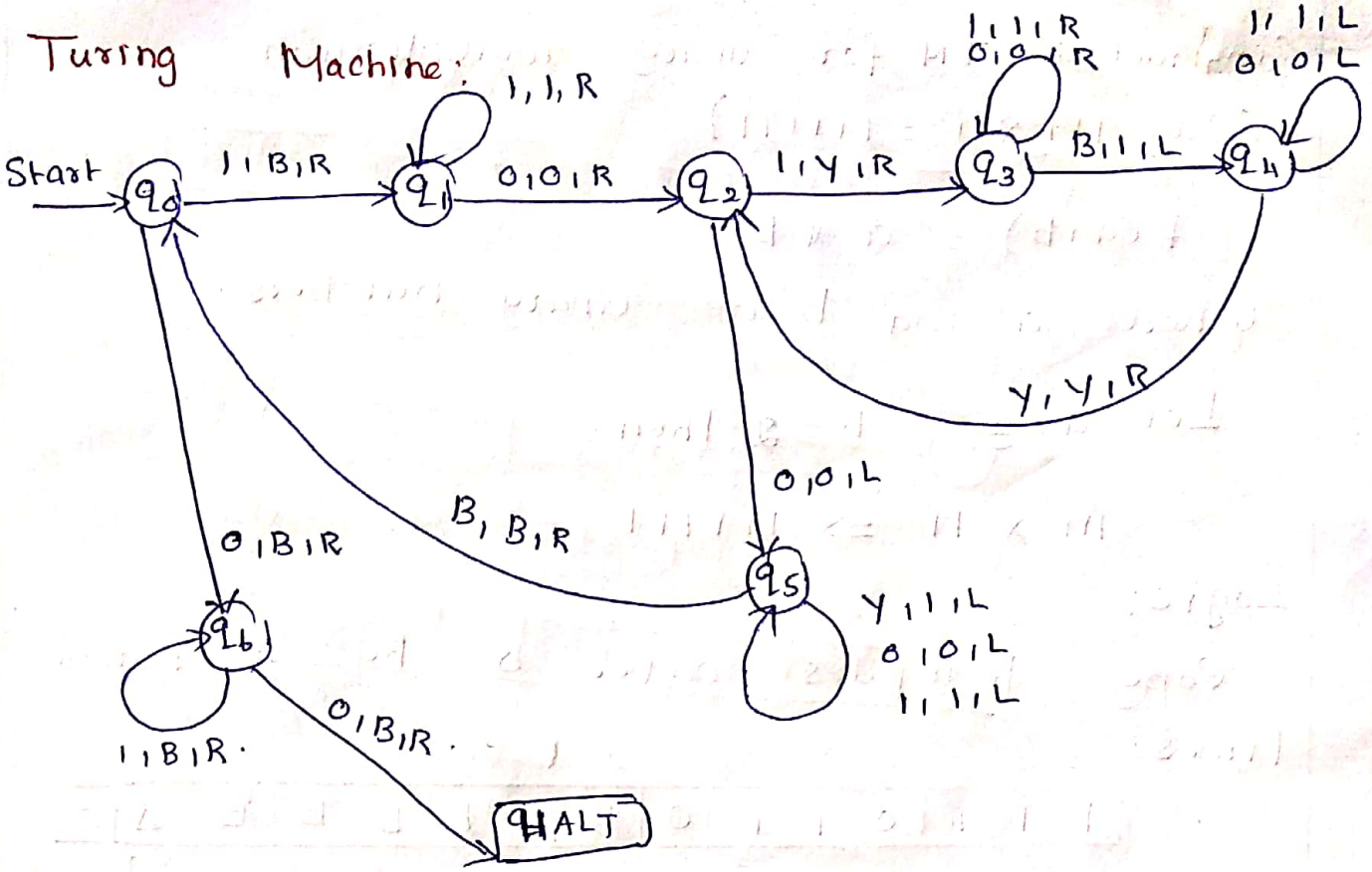
8) B 1 1 0 Y 1 0 B B B B B B Δ
↑ L

9) B 1 1 0 Y 1 0 1 B B B B B Δ
⋮

10. B B B 0 Y Y 0 1 1 1 1 1 1 Δ.

11. Repeat the Step up to B B B B B B B 1 1 1 1 1 1

Turing Machine:



Transition Table:

	δ	0	1	Y	B
→	q_0	q_6 B R	q_1 B R	–	–
	q_1	q_2 0 R	q_1 1 R	–	–
	q_2	q_5 0 L	q_3 Y R	–	–
	q_3	q_3 0 R	q_3 1 R	–	q_4 1 L
	q_4	q_4 0 L	q_4 1 L	q_2 Y R	
	q_5	q_5 0 L	q_5 1 L	q_5 1 L	q_0 B R
	q_6	q_{Halt} B R	q_6 B R	–	–
	HALT	–	–	–	–

Multihead and Multi tape Turing Machine:

Multitape Turing Machine:

This kind of TM that have one finite control and more than one tapes each with its own read-write head. It is donoted by a 5-tuple as

$$\langle Q, \Sigma, \Gamma, q_0, \delta \rangle$$

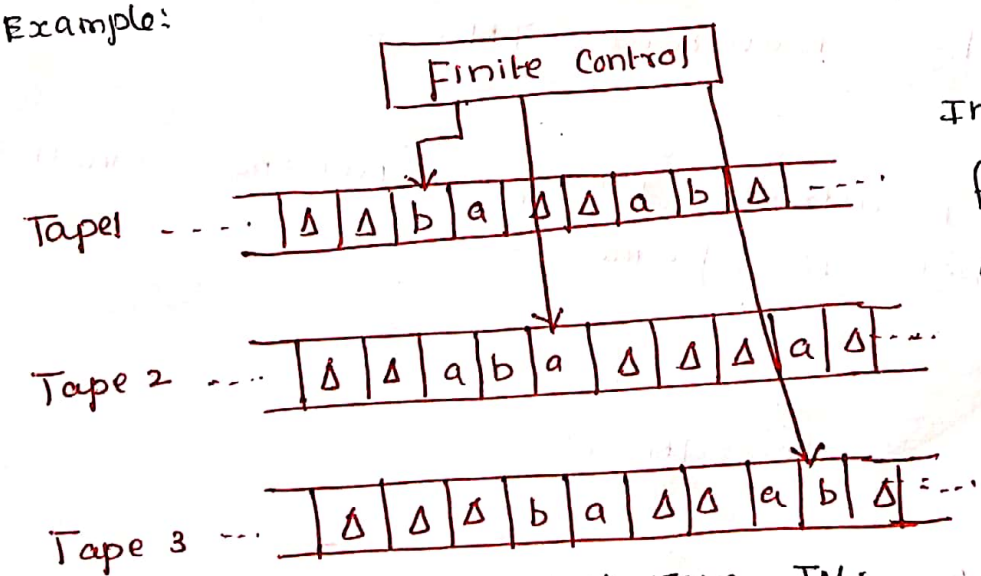
Its transition function is a partial function.

$$\delta : Q \times (\Gamma \cup \{\Delta\})^n \to (Q \cup \{h\} \times \{\Gamma \cup \{\Delta\}\})^n \times \{R, L, S\}^n$$

A configuration for this kind of TM must show the current state the machine is in and the state of each tape.

It can be proven that any language accepted by an n-tape Turing machine can be accepted by a one tape TM and that any function computed by an n-tape TM can be computed by a one tape TM. One tape TM are as powerful as n-tape TM.

Example:



Input can be read from more than one input tape.

Fig: Multi Tape TM.

2. Multihead TM:

This kind of TM that have one finite control and one tape but more than one read-write heads. In each state only one of the heads is allowed to read and write.

It is denoted by a 5-tuple

$$\langle Q, \delta : Q \times \{H_1, H_2, H_3 \cdots H_n\} \times (\Gamma \cup \{\Delta\}) \rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{R, L, S\}$$

where $H_1, H_2, \cdots H_n$ denote the tape heads.

It can be easily seen that this type of TM are as powerful as one tape TM.

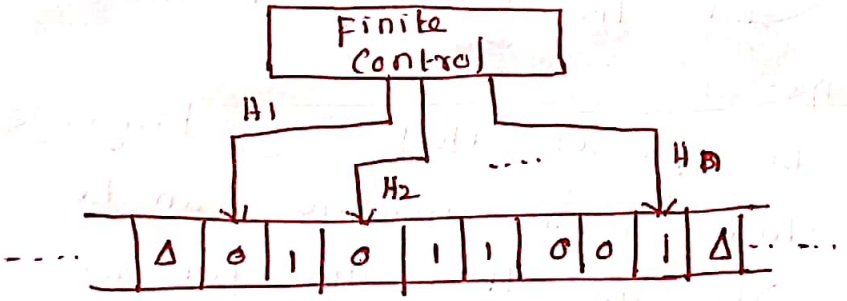


Fig. Multihead TM.

More than one input symbol can be read at a time by multihead TM.

Example:

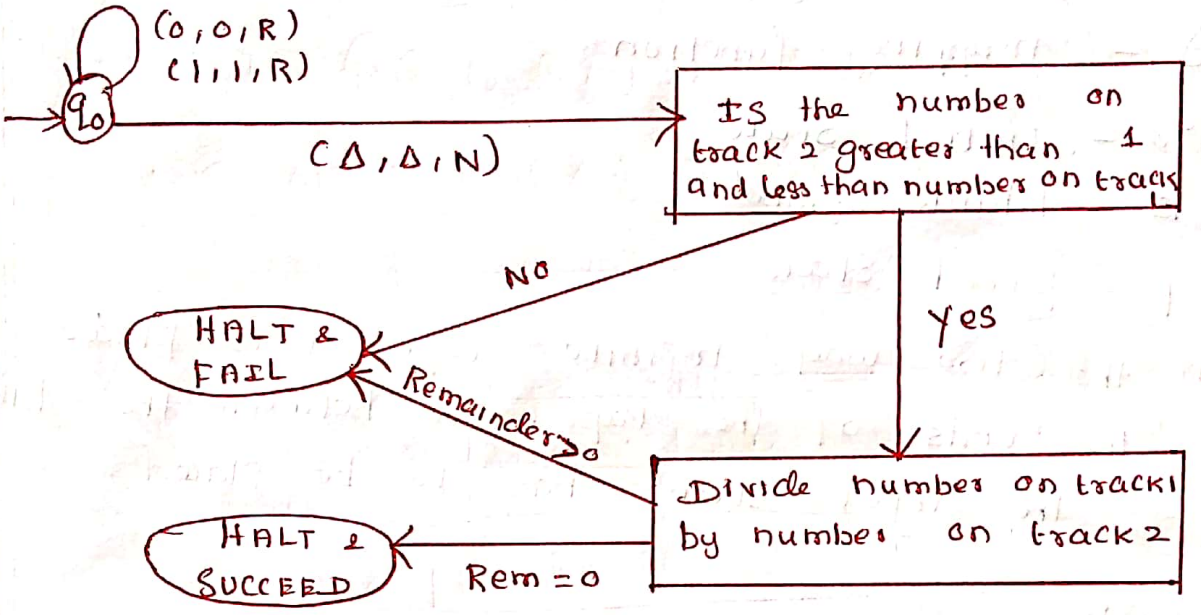
Build Multitrack TM for checking whether the given number is prime (or) not.

Solution:

Logic for the TM is

1. Let the number M, where $1 < M < n$
2. Divide n by M.
3) If there is a remainder then it halts and succeed.
4. Otherwise it halts and fails.

It can be modelled as.



Two way infinite Tape:-

The TM is widely accepted as a model of computation because of its versatality in the power of computation. The TM can perform string operations, arithmetic computations, recognizing languages either regular or non regular languages.

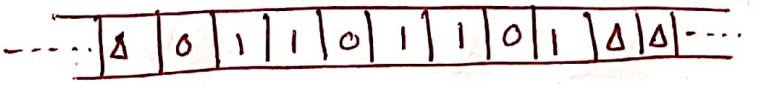


Fig: Two way infinite Tape.

The tape head as usual can move in forward and backward direction.

The TM with infinite tape can also be denoted by

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

where,

Q - finite non empty set of states.

Σ - Nonempty finite set of input

Γ - Set of external symbol used.

δ - mapping function.

q_0 - Initial state.

B - Blank Symbol

F - Final state.

In the two way infinite tape it is placed at both the ends of the tape. In between the blank Symbols the input string has to be placed.

Theorem:

L is recognized by a TM with a two way infinite tape if and only if it is recognized by a TM with one way infinite tape.

Proof:

As theorem states, if any language is recognized by a TM with one way infinite tape then it should also be recognized by a TM with two way infinite tape.

Let M_1 be a TM with one way infinite tape and can be denoted by

$$M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_1, B, F_1)$$

Similarly, M_2 be a TM with two way infinite tape and can be denoted by

$$M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, S_1, B, F_2)$$

The input tapes are as shown by following figure.

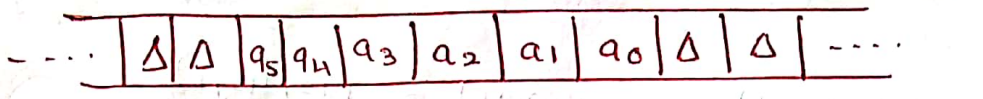


Fig: Two way infinite tape for M_2 TM

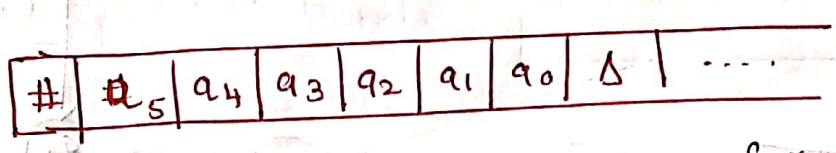
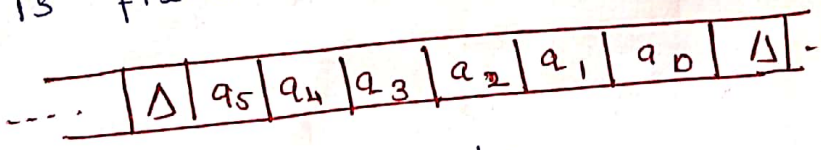


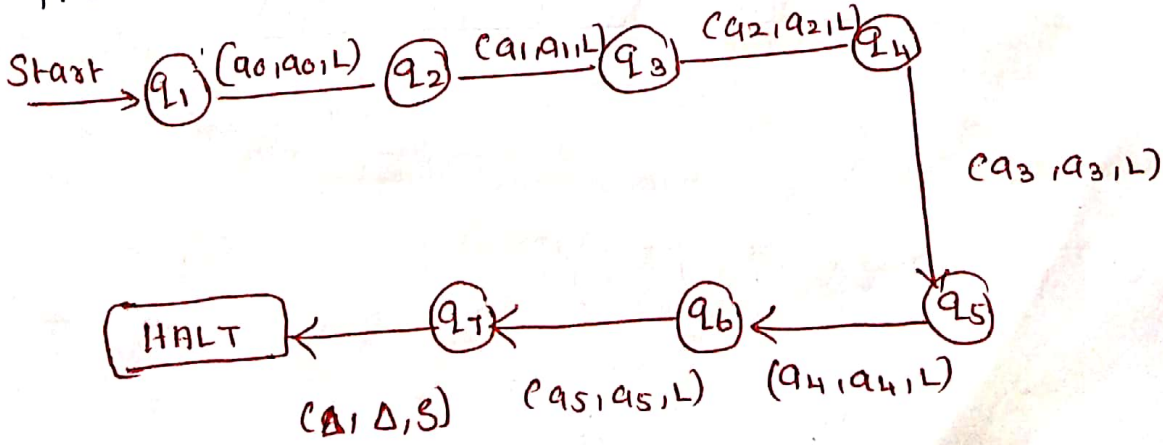
Fig: One way infinite tape for M_1 TM.

As Shown in the above figure, the TM with one way infinite tape has a external Symbol # placed in the very first cell from left side. This Symbol is used as indicator for the left side termination.

If we want a Language $L = \{q_0, q_1, q_2, q_3, q_4, q_5\}$ Sequence then in the two way infinite tape the tape head is fixed at the rightmost Symbol.

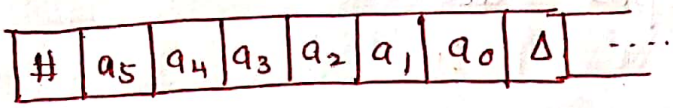


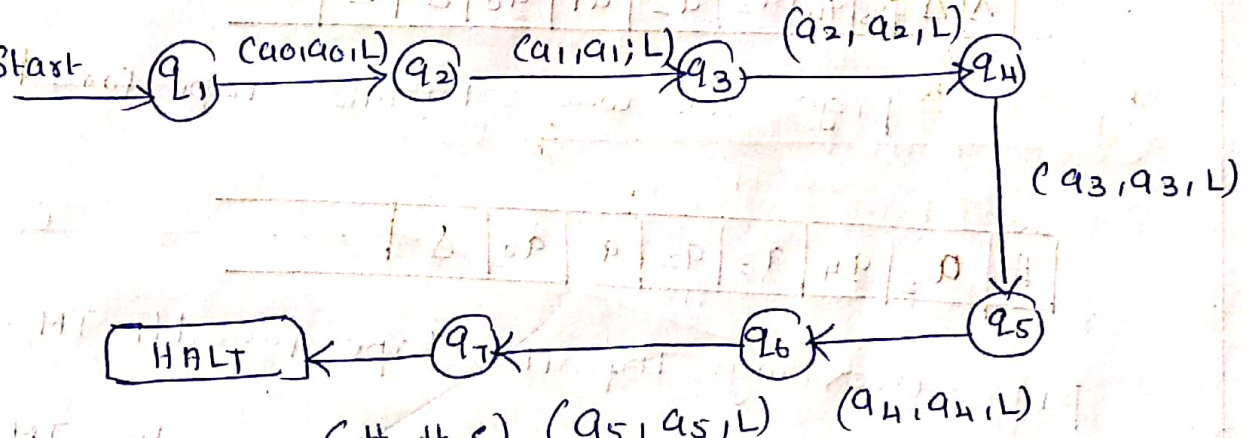
The TM M_2 can be.



$\delta(q_7, \Delta) \rightarrow (HALT, \Delta, S)$ leads to HALT state.

Similarly, with one way infinite tape the machine M_1 will be.





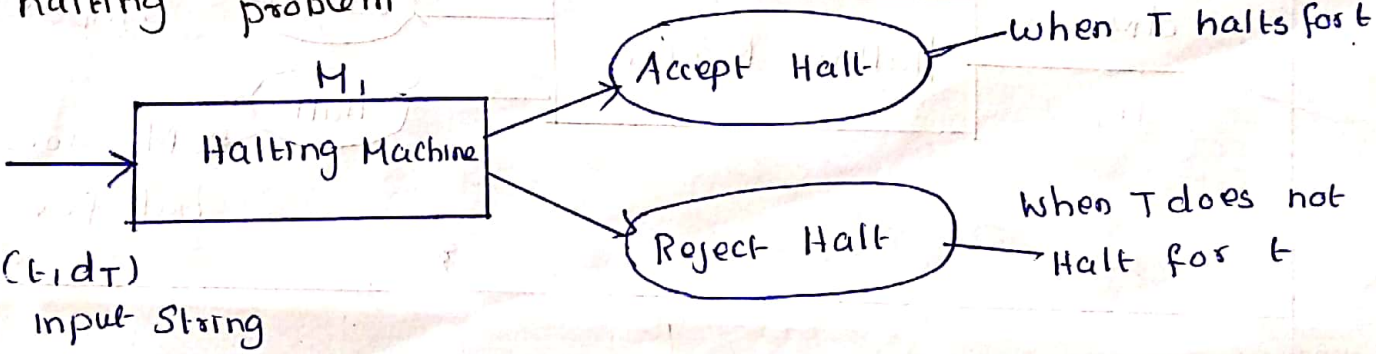
Even we can make the TM with one way infinite tape as a multitrack tape to simulate it as a two way infinite tape.

Halting Problems:

The output of TM can be

1. **HALT:** The TM starting at this configuration will halt after a finite number of states.
2. **NO HALT:** The TM starting at this configuration never reaches a halt state.

Given any functional matrix, input data tape and initial configuration, then is it possible to determine whether the process will halt is called halting problem.

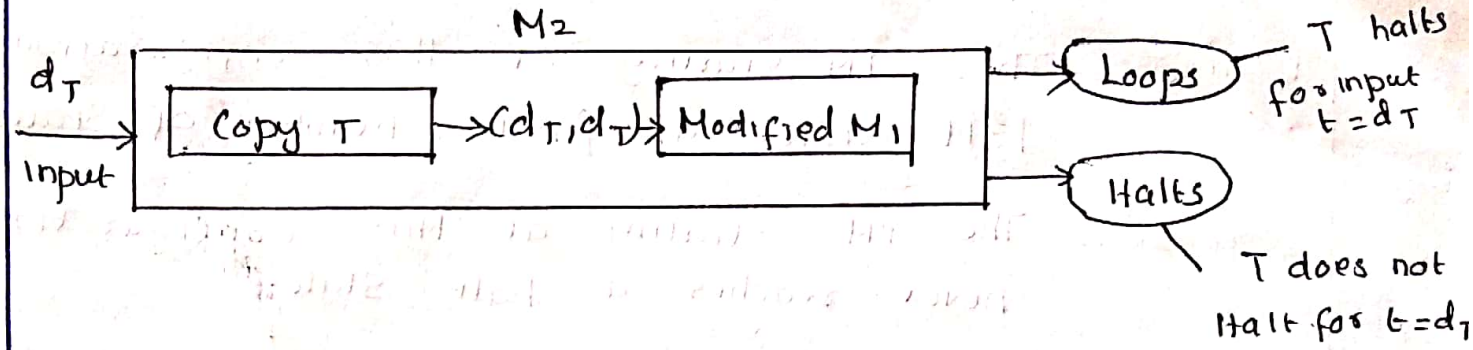


For every input (t, d_T) to M_1, if T halts for input t, M_1 also halts which is called Accept halt. Similarly if T does not halt for input t, then the M_1 will halt which is called Reject Halt.

Now consider another Turing Machine M_2 which takes an input d_T, It first copies d_T and duplicates d_T on its tape and then this duplicated tape information is given as input to machine M_1, But M_1 is modified machine that whenever M_1 is supposed to reach an accept halt, M_1 loops forever. Hence the behaviour of M_2 is.

1. It loops if T halts for input $t = d_T$.

2. It halts if T does not halt for $T = d_T$



Assume M_2 itself one Turning Machine is $T = M_2$. That means replace T by M_2.