

**SRI CHANDRASEKHARENDRA SARASWATHI VISVA
MAHAVIDYALAYA**

(Deemed to be University under sec. 3 of UGC Act.1956)

(Accredited with “A” Grade by NAAC)

ENATHUR, KANCHIPURAM-631561.



Lecture Notes

Prepared by Dr. T. N. Kavitha

Automata Theory – Unit V

Turing Machine

Subject Code: PCC-CS501

Course : B.E (CSE)

UNIT - V- TURING MACHINE

INTRODUCTION TO TURING MACHINES

Problems that computers cannot solve – The Turing machine – Programming techniques for Turing machines – Extensions to the basic Turing machine – Restricted Turing machines – Turing machines and computers

LECTURE PLAN

(INTRODUCTION TO TURING MACHINES)
Introduction
The Turing machine
Programming techniques for Turing machines
Extensions to the basic Turing machine
Restricted Turing machines
Turing machines and computers
Unit-V test

Turing Machine Introduction

A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

Definition

A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected.

A TM can be formally described as a 7-tuple $(Q, X, \Sigma, \delta, q_0, B, F)$ where –

- **Q** is a finite set of states
- **X** is the tape alphabet
- Σ is the input alphabet
- δ is a transition function; $\delta : Q \times X \rightarrow Q \times X \times \{\text{Left_shift}, \text{Right_shift}\}$.
- **q₀** is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Comparison with the previous automation

The following table shows a comparison of how a Turing machine differs from Finite Automaton and Pushdown Automaton.

Machine	Stack Data Structure	Deterministic?
Finite Automaton	N.A	Yes
Pushdown Automaton	Last In First Out(LIFO)	No
Turing Machine	Infinite tape	Yes

Example of Turing machine

Turing machine $M = (Q, X, \Sigma, \delta, q_0, B, F)$ with

- $Q = \{q_0, q_1, q_2, q_f\}$
- $X = \{a, b\}$
- $\Sigma = \{1\}$
- $q_0 = \{q_0\}$
- $B = \text{blank symbol}$
- $F = \{q_f\}$
- δ is given by –

Tape alphabet symbol Present State ‘ q_0 ’ Present State ‘ q_1 ’ Present State ‘ q_2 ’

a	1R q_1	1L q_0	1L q_f
b	1L q_2	1R q_1	1R q_f

Here the transition 1R q_1 implies that the write symbol is 1, the tape moves right, and the next state is q_1 . Similarly, the transition 1L q_2 implies that the write symbol is 1, the tape moves left, and the next state is q_2 .

Time and Space Complexity of a Turing Machine

For a Turing machine, the time complexity refers to the measure of the number of times the tape moves when the machine is initialized for some input symbols and the space complexity is the number of cells of the tape written.

Time complexity all reasonable functions –

$$T(n) = O(n \log n)$$

TM's space complexity – $S(n) = O(n)$

Accepted Language & Decided Language

A TM accepts a language if it enters into a final state for any input string w . A language is recursively enumerable (generated by Type-0 grammar) if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is recursive if it is decided by a Turing machine.

There may be some cases where a TM does not stop. Such TM accepts the language, but it does not decide it.

Designing a Turing Machine

The basic guidelines of designing a Turing machine have been explained below with the help of a couple of examples.

Example 1

Design a TM to recognize all strings consisting of an odd number of a 's.

Solution

The Turing machine **M** can be constructed by the following moves –

- Let q_1 be the initial state.
- If **M** is in q_1 ; on scanning a , it enters the state q_2 and writes **B** (blank).
- If **M** is in q_2 ; on scanning a , it enters the state q_1 and writes **B** (blank).
- From the above moves, we can see that **M** enters the state q_1 if it scans an even number of a 's, and it enters the state q_2 if it scans an odd number of a 's. Hence q_2 is the only accepting state.

Hence,

$$M = \{\{q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_1, B, \{q_2\}\}$$

where δ is given by –

Tape alphabet symbol Present State ' q_1 ' Present State ' q_2 '

a

BRq_2

BRq_1

Example 2

Design a Turing Machine that reads a string representing a binary number and erases all leading 0's in the string. However, if the string comprises of only 0's, it keeps one 0.

Solution

Let us assume that the input string is terminated by a blank symbol, B, at each end of the string.

The Turing Machine, **M**, can be constructed by the following moves –

- Let q_0 be the initial state.
- If **M** is in q_0 , on reading 0, it moves right, enters the state q_1 and erases 0. On reading 1, it enters the state q_2 and moves right.

- If **M** is in **q₁**, on reading 0, it moves right and erases 0, i.e., it replaces 0's by B's. On reaching the leftmost 1, it enters **q₂** and moves right. If it reaches B, i.e., the string comprises of only 0's, it moves left and enters the state **q₃**.
- If **M** is in **q₂**, on reading either 0 or 1, it moves right. On reaching B, it moves left and enters the state **q₄**. This validates that the string comprises only of 0's and 1's.
- If **M** is in **q₃**, it replaces B by 0, moves left and reaches the final state **q_f**.
- If **M** is in **q₄**, on reading either 0 or 1, it moves left. On reaching the beginning of the string, i.e., when it reads B, it reaches the final state **q_f**.

Hence,

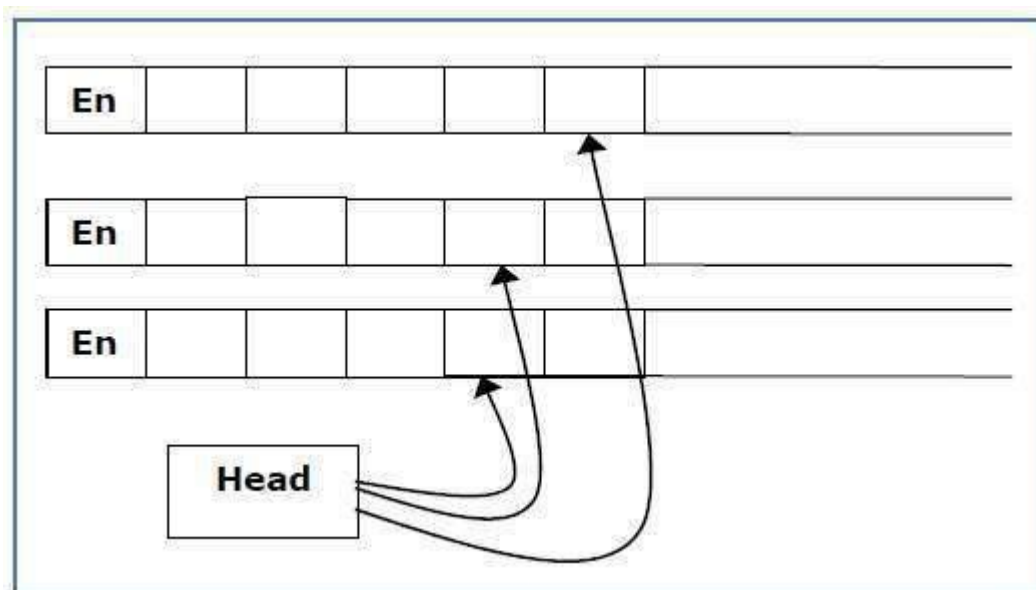
$$M = \{\{q_0, q_1, q_2, q_3, q_4, q_f\}, \{0, 1, B\}, \{1, B\}, \delta, q_0, B, \{q_f\}\}$$

where δ is given by –

Tape alphabet symbol	Present State 'q ₀ '	Present State 'q ₁ '	Present State 'q ₂ '	Present State 'q ₃ '	Present State 'q ₄ '
0	BRq ₁	BRq ₁	ORq ₂	-	OLq ₄
1	1Rq ₂	1Rq ₂	1Rq ₂	-	1Lq ₄
B	BRq ₁	BLq ₃	BLq ₄	OLq _f	BRq _f

Multi-tape Turing Machine

Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



A Multi-tape Turing machine can be formally described as a 6-tuple $(Q, X, B, \delta, q_0, F)$ where –

- **Q** is a finite set of states
- **X** is the tape alphabet
- **B** is the blank symbol
- δ is a relation on states and symbols where

$$\delta: Q \times X^k \rightarrow Q \times (X \times \{\text{Left_shift}, \text{Right_shift}, \text{No_shift}\})^k$$

where there is **k** number of tapes

- **q₀** is the initial state
- **F** is the set of final states

Note – Every Multi-tape Turing machine has an equivalent single-tape Turing machine.

Multi-track Turing Machine

Multi-track Turing machines, a specific type of Multi-tape Turing machine, contain multiple tracks but just one tape head reads and writes on all tracks. Here, a single tape head reads **n** symbols from **n** tracks at one step. It accepts recursively enumerable languages like a normal single-track single-tape Turing Machine accepts.

A Multi-track Turing machine can be formally described as a 6-tuple (Q, X, Σ , δ , q₀, F) where –

- **Q** is a finite set of states
- **X** is the tape alphabet
- Σ is the input alphabet
- δ is a relation on states and symbols where

$$\delta(Q_i, [a_1, a_2, a_3, \dots]) = (Q_j, [b_1, b_2, b_3, \dots], \text{Left_shift or Right_shift})$$

- **q₀** is the initial state
- **F** is the set of final states

Note – For every single-track Turing Machine **S**, there is an equivalent multi-track Turing Machine **M** such that **L(S) = L(M)**.

Non-Deterministic Turing Machine

In a Non-Deterministic Turing Machine, for every state and symbol, there are a group of actions the TM can have. So, here the transitions are not deterministic. The computation of a non-deterministic Turing Machine is a tree of configurations that can be reached from the start configuration.

An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted. If all branches of the computational tree

halt on all inputs, the non-deterministic Turing Machine is called a **Decider** and if for some input, all branches are rejected, the input is also rejected.

A non-deterministic Turing machine can be formally defined as a 6-tuple $(Q, X, \Sigma, \delta, q_0, B, F)$ where –

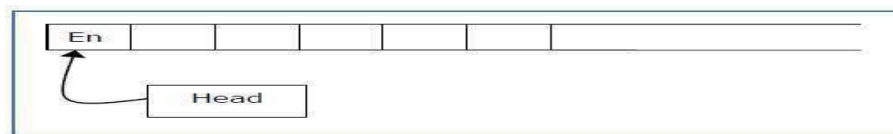
- **Q** is a finite set of states
- **X** is the tape alphabet
- Σ is the input alphabet
- δ is a transition function;

$$\delta : Q \times X \rightarrow P(Q \times X \times \{\text{Left_shift}, \text{Right_shift}\}).$$

- q_0 is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Semi-Infinite Tape Turing Machine

A Turing Machine with a semi-infinite tape has a left end but no right end. The left end is limited with an end marker



It is a two-track tape –

- **Upper track** – It represents the cells to the right of the initial head position.
- **Lower track** – It represents the cells to the left of the initial head position in reverse order.

The infinite length input string is initially written on the tape in contiguous tape cells.

The machine starts from the initial state q_0 and the head scans from the left end marker 'End'. In each step, it reads the symbol on the tape under its head. It writes a new symbol on that tape cell and then it moves the head either into left or right one tape cell. A transition function determines the actions to be taken.

It has two special states called **accept state** and **reject state**. If at any point of time it enters into the accepted state, the input is accepted and if it enters into the reject state, the input is rejected by the TM. In some cases, it continues to run infinitely without being accepted or rejected for some certain input symbols.

Note – Turing machines with semi-infinite tape are equivalent to standard Turing machines.

QUESTION AND ANSWER

1. Differentiate Turing Machine and Pushdown Automata.

Turing Machine is modified version of the PDA. TM is more powerful than PDA.

Instead of using stack in PDA, the TM uses the tape to store the symbols. TM can accept any language.

2. What are the components in TM?
Tape, Read-write head and control unit.

3. Define Turing machine.

The Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where

Q : Set of finite states

Σ : Set of input alphabets

Γ : Set of tape symbols

δ : transition from $Q \times \Gamma$ to $Q \times \Gamma \times (L, R, N)$

B : Blank character

q_0 : Start state

$F \subseteq Q$ set of final state

The TM can be called Standard Turing Machine with the following features”

The TM has a tap that is divided into number of cells with each cell capable of storing only one symbol. The tape is unbounded with any number of left or right moves.

4. Brief about Turing machines as language acceptors.

A language L is recursively enumerable, if it is accepted by a TM, i.e., given a string w which is input to TM, the machine halts and outputs Yes if it belongs to the language. If w does not belong to the language L , the TM halts and outputs NO.

5. Define move of a TM.

Possible different types of moves are

- Moves to left
- Moves to right
- No move.

6. What language is accepted by TM?

The TM is a generalized machine which can recognize all types of languages i.e., context free language, regular language and unrestricted language.

7. What is an ID with respect to TM?

In TM the ID(Instantaneous Description) is defined on the whole string and the current state of machine .An ID of TM is a string in $\alpha q \beta$ where q is the start state and the read –write head points to the first symbol of α from left. The final ID is denoted by $\alpha \beta q B$ where $q \in F$ is the final state and read – write head points to the blank character denoted by B

8. Define Non-deterministic Turing Machine.

Non-deterministic Turing Machine has 7 tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, h)$

where

Q: Set of finite states

Σ : Set of non-blank symbols

Γ : Set of tape symbols

B: Blank character

q_0 : Start state

$h \subseteq Q$ set of final state

$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times (L, R, N))$

9. Brief about Linear bounded automata.

The Linear bounded Automaton is a TM

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

Q: Set of finite states

Σ : Set of input alphabets which also has two special symbols '[' and '']

Γ : Set of tape symbols

B: Blank character

q_0 : Start state

$F \subseteq Q$ set of final state

$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times (L, R)}$ with two more transition of the form

$\delta(q_i, [) = (q_i, [, R)$ and $\delta(q_i,]) = (q_i,], R)$ forcing the read /write head to be within the boundaries '[' and '']'.

10. Define universal TM

A universal TM is an automaton that, given as input the description of any TM, M and a string w, can simulate the computation of M on w Assume that

$Q = \{q_1, q_2, \dots, q_n\}$ where q_1 is the initial state and q_n is the final state

11. What do you mean by a TM with stay-option?

Apart from having the read/write either left or right, if there is one more option where in the read / write head stays in the same position after updating the symbol on the tape, then the Turing Machine is called TM with stay-option.

12. Define TM with stay-option

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

Q : Set of finite states

Σ : Set of input alphabets

Γ : Set of tape symbols

δ : transition from $Q \times \Gamma$ to $Q \times \Gamma \times (L, R, S)$ indicating the TM may move towards left or right or stay in the same position after updating the symbol on the tape.

B : Blank character

q_0 : Start state

$F \subseteq Q$ set of final state

13. When does a TM halt?

If there is no entry in the table for the current combination of symbol and state, then the machine will halt.

14. Which language is accepted by counter machines?

Context free language is accepted by counter machines.

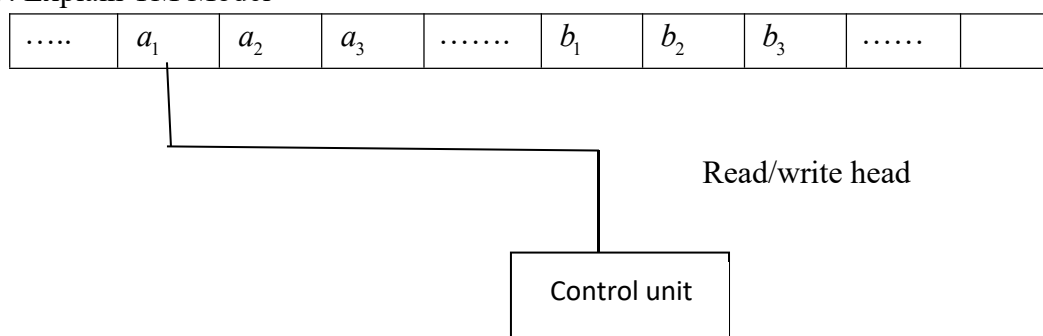
15. What is TM computable?

The arithmetic operations such as addition, subtraction etc., including common mathematical functions are Turing computable.

16. On what basis the TM is a transducer?

A transducer accepts some input and transforms the input into the desired output. In this sense, the TM can be called as a transducer.

17. Explain TM Model



18. What is a multi-tape TM?

A multi-tape TM is a standard TM with more number of tapes. Each tape is controlled independently with independent read-write head.

19. Define Off-line TM

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

Q : Set of finite states

Σ : Set of input alphabets

Γ : Set of tape symbols

δ : Transition from $Q \times \Sigma \times \Gamma$ to $Q \times \Gamma \times (L, R)$

B : Blank character

q_0 : Start state

$F \subseteq Q$ set of final state

20. Define TM with multiple tracks

Divide the tape into number of tracks and each track is divided into a number of squares with each square holding only one symbol. If there are n -tracks and the read-write head points to m^{th} square,

Then all symbols under different tracks that read/write head are the symbols to be scanned. So, all the symbols under the read/write head can be considered as set of characters under one square. Such a TM is called machine with multiple tracks.

UNIT V- PROBLEMS FOR PRACTICE- TURING MACHINE

1. Obtain a TM to accept the language $L = \{w / w \in (0+1)^*\}$ containing the substring 001.

Solution

Step 1 : DFA consists of 0's & 1's having substring 001

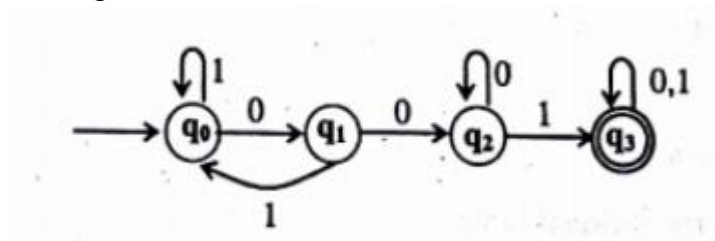
Step 2: Draw Transition diagram

Step 3: Transition table -1

Step 4 : Transition table -2

Step 5: Turing Machine

The DFA which accepts the language consisting of strings of 0's and 1's having a sub string 001 is shown below:



The transition table

	0	1
--	---	---

q ₀	q ₁	q ₀
q ₁	q ₂	q ₀
q ₂	q ₂	q ₃
q ₃	q ₃	q ₃

We have seen already in previous unit that any language which is accepted by a DFA is regular. As the DFA processes the input string from left to right in only one direction,

TM also processes the input string in only one direction.

For each scanned input symbol (either 0 or 1), in which ever state the DFA was in, TM also enters into the same states on same input symbols, replacing 0 by 0 and 1 by 1 and the read- write head moves towards right.

So the transition table for DFA and TM remains the same.

So the transition table for TM to recognize the language consisting of 0's and 1's with a substring 001 is shown below :

	0	1	B
q ₀	q ₁ ,0,R	q ₀ ,1,R	-
q ₁	q ₂ ,0,R	q ₀ ,1,R	-
q ₂	q ₂ ,0,R	q ₃ ,1,R	-
q ₃	q ₃ ,0,R	q ₃ ,1,R	q ₄ ,B,R
q ₄			

The TM is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where $Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$

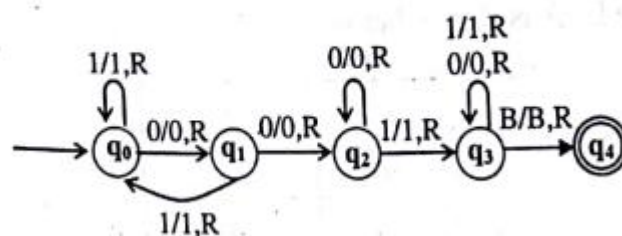
δ is shown in the form of transition table above

q₀ is the start state

B blank character

$F = \{q_4\}$ is the final state

The transition diagram for this is shown below:



2. Obtain a TM to accept the language $L = \{w / w \text{ is even and } \Sigma = \{a, b\}\}$

Step 1: DFA consists of even number of characters

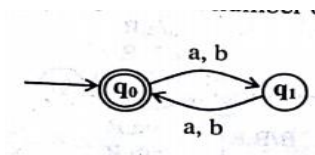
Step 2: Transition diagram

Step 3: Transition table -1

Step 4: Transition table -2

Step 5:Turing Machine

The DFA to accept the language consisting of even number of characters is shown below:



The transition table for the DFA is shown below:

	a	b
q ₀	q ₁	q ₁
q ₁	q ₀	q ₀

We have seen in previous unit that any language which is accepted by a DFA is regular. As the DFA processes the input string from left to right in only one direction, TM also processes the input string in only one direction. For each scanned input symbol (either a or b), in whichever state the DFA was in, TM also enters into the same states on same input symbols, replacing a by a and b by b and the read- write head moves towards right. So, the transition table for DFA and TM remains same(the format may be different). So, the transition table for TM to recognize the language consisting of a's and b's having even number of symbols shown below:

δ	A	b	B
q ₀	q ₁ ,a,R	q ₁ ,b,R	q ₂ ,B,R
q ₁	q ₀ ,a,R	q ₀ ,b,R	-
q ₂	-	-	-

The TM is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where $Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, B\}$

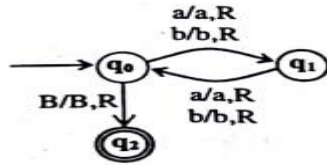
δ is shown in the form of transition table above

q₀ is the start state

B blank character

$F = \{q_2\}$ is the final state

The transition diagram for TM is shown below:



3. Obtain a TM to accept a palindrome consisting of a's and b's of any length.

SOLUTION

Step 1: Transition of the pointer to be in the first character of the string

Step 2: Transition diagram

Step 3: Pointer to be in the next state

Step 4: Tm accept strings of palindromes over {a,b}

Step 5: Transition table

Let us assume that the first symbol on the tape is blank character B and is followed by the string which in turn ends with blank character B. Now, we have to design a Turing machine which accepts the string, provided the string is a palindrome. For the string to be a palindrome, the first and the last character should be same.

The second character and last but one character should be same and so on. The procedure to accept only string of palindromes is shown below. Let q_0 be the start state of Turing machine.

Step 1 :

Move the read- write head to point to the first character of the string. The transition for this can be of the form

$$\delta(q_0, B) = (q_1, B, R)$$

Step 2:

In state q_1 , if the first character is a, replace it by B and change the state to q_2 , and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1, a) = (q_2, B, R)$$

Now, we move the read- write head to point to the last symbol of the string and the last symbol should be a. The symbols scanned during this process are a's and b's and B. Replace a by a, b by b and move the pointer towards right. The transitions defined for this can be of the form

$$\delta(q_2, a) = (q_2, a, R)$$

$$\delta(q_2, B) = (q_3, B, R)$$

But, one the symbol B is encountered, change the state to q_3 , replace B by B and move the pointer towards left. The transition defined for this can be of the form

$$\delta(q_3, B) = (q_3, B, L)$$

In state q_3 , the read-write head points to the last character of the string. If the last character is a, then change the state to q_4 , replace a by B and move the pointer towards left. The transitions defined for this can be the form.

$$\delta(q_3, a) = (q_4, B, L)$$

At this point, we know that the first character is a and the last character is also a. now, reset the read-write head to point to the first non blank character as shown in step 5.

In state q_3 , if the last character is B (blank character), it means that the given string is an odd palindrome. So, replace B by B change the state to q_7 move the pointer towards right. The transition for this can be of the form

$$\delta(q_3, B) = (q_7, B, R)$$

Step 3:

If the first character is the symbol b, replace it by B and change the state from q_1 to q_5 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1, b) = (q_5, B, R)$$

Now, we move the read-write head to point to the last symbol of the string and the last symbol should be b. the symbols scanned during this process are a's, b's and B. Replace a by a, b by b and move the pointer towards right. The transition defined for this can be of the form

$$\delta(q_5, a) = (q_5, a, R)$$

$$\delta(q_5, b) = (q_5, b, R)$$

But, once the symbol B is encountered, change the state to q_6 , replace B by B and move the pointer towards left, the transition defined for this can be of the form

$$\delta(q_6, b) = (q_4, B, L)$$

In state q_6 , the read-write head points to the last character of the string. If the last character is b, change the state to q_6 , replace b by B and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_6, b) = (q_4, B, L)$$

At this point, we know that the first character is b and last character is also b. Now, reset the read-write head to point to the first non blank character as shown in step 5.

In state q_3 , if the last character is B (blank character), it means that the given string is an odd palindrome. So, replace B by B change the state to q_7 move the pointer towards right. The transition for this can be of the form

$$\delta(q_3, B) = (q_7, B, R)$$

Step 3:

If the first character is the symbol b, replace it by B and change the state from q_1 to q_5 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1, b) = (q_5, B, R)$$

Now, we move the read-write head to point to the last symbol of the string and the last symbol should be b. the symbols scanned during this process are a's, b's and B. Replace a by a, b by b and move the pointer towards right. The transition defined for this can be of the form

$$\delta(q_5, a) = (q_5, a, R)$$

$$\delta(q_5, b) = (q_5, b, R)$$

But, once the symbol B is encountered, change the state to q_6 , replace B by B and move the pointer towards left, the transition defined for this can be of the form

$$\delta(q_6, b) = (q_4, B, L)$$

In state q_6 , the read-write head points to the last character of the string. If the last character is b, the change the state to q_6 , replace b by B and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_6, b) = (q_4, B, L)$$

At this point, we know that the first character is b and last character is also b. Now, reset the read- write head to point to the first non blank character as shown in **Step 5**.

In state q_6 , If the last character is B(blank character), it means that the given string is an odd palindrome. So, replace B by B, change the state to q_7 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_6, B) = (q_7, B, R)$$

Step 4:

In state q_1 , if the first symbol is blank character(B), the given string is even palindrome and so change the state to q_7 , replace B by B and move the read-write head towards right. The transition for this can be of the form

$$\delta(q_1, B) = (q_7, B, R)$$

Step 5:

Reset the read-write head to point to the first non blank character. This can be done as down below. If the first symbol of the string is a, step 2 is performed and if the first symbol of the string is b, Step 3 is performed. After completion of step 2 or step 3, it is cleat that the first

Symbol and the last symbol match and the machine is currently in state q_4 . Now, we have to reset the read- write head to point to the first non blank character in the string by repeatedly moving the head towards left and remain in state q_4 . During this process, the symbols encountered may be a or b or B (blank character). Replace a by a, b by b and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_4, a) = (q_4, a, L)$$

$$\delta(q_4, b) = (q_4, b, L)$$

But, if the symbol B ius encountered, change the state to q_1 , replace B by B and move the pointer towards right. The transition defined for this can be of the form

$$\delta(q_4, B) = (q_1, B, R)$$

After resetting the read-write head to the first non-blank character,

repeat through step 1. so, the TM to accept strings of palindromes over $\{a,b\}$ is give by

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

where

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{a, b, B\}$$

q_0 is the start state

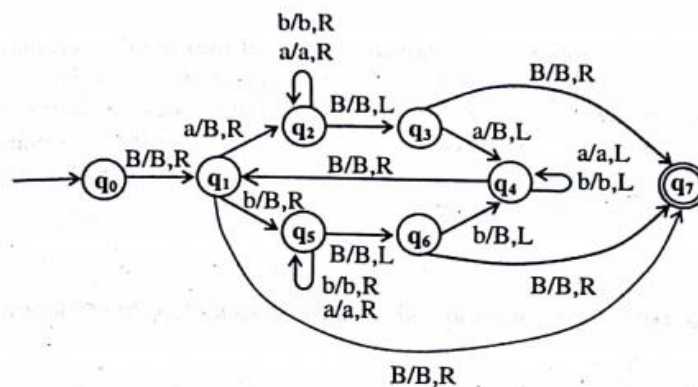
B is the blank character

$$F = \{q_7\}$$

δ is shown below using the transition table:

	δ	Γ		
		a	b	B
→	q_0	-	-	q_1, B, R
	q_1	q_2, B, R	q_5, B, R	q_7, B, R
	q_2	q_2, a, R	q_2, b, R	q_3, B, L
	q_3	q_4, B, L	-	q_7, B, R
	q_4	q_4, a, L	q_4, b, L	q_1, B, R
	q_5	q_5, a, R	q_5, b, R	q_6, B, L
	q_6	-	q_4, B, L	q_7, B, R
	$*q_7$	-	-	-

The transition diagram to accept palindromes over $\{a, b\}$ is given by



The reader can trace the moves made by the machine for the strings abba, aba and aaba and is left as an exercise.

4. Design a TM to recognize all strings consisting of an odd number of a 's.

Solution

The Turing machine **M** can be constructed by the following moves –

Let **q1** be the initial state.

If **M** is in **q1**; on scanning α , it enters the state **q2** and writes **B** (blank).

If **M** is in **q2**; on scanning α , it enters the state **q1** and writes **B** (blank).

From the above moves, we can see that **M** enters the state **q1** if it scans an even number of α 's, and it enters the state **q2** if it scans an odd number of α 's. Hence **q2** is the only accepting state.

Hence,

$M = \{\{q1, q2\}, \{1\}, \{1, B\}, \delta, q1, B, \{q2\}\}$

where δ is given by –

Tape alphabet symbol	Present State 'q1'	Present State 'q2'
α	BRq2	BRq1

5. Design a Turing Machine that reads a string representing a binary number and erases all leading 0's in the string. However, if the string comprises of only 0's, it keeps one 0.

Solution

Let us assume that the input string is terminated by a blank symbol, B, at each end of the string.

The Turing Machine, **M**, can be constructed by the following moves –

Let **q0** be the initial state.

If **M** is in **q0**, on reading 0, it moves right, enters the state **q1** and erases 0. On reading 1, it enters the state **q2** and moves right.

If **M** is in **q1**, on reading 0, it moves right and erases 0, i.e., it replaces 0's by B's. On reaching the leftmost 1, it enters **q2** and moves right. If it reaches B, i.e., the string comprises of only 0's, it moves left and enters the state **q3**.

If **M** is in **q2**, on reading either 0 or 1, it moves right. On reaching B, it moves left and enters the state **q4**. This validates that the string comprises only of 0's and 1's.

If **M** is in **q3**, it replaces B by 0, moves left and reaches the final state **qf**.

If **M** is in **q4**, on reading either 0 or 1, it moves left. On reaching the beginning of the string, i.e., when it reads B, it reaches the final state **qf**.

Hence,

$M = \{\{q0, q1, q2, q3, q4, qf\}, \{0, 1, B\}, \{1, B\}, \delta, q0, B, \{qf\}\}$

where δ is given by –

Tape alphabet symbol	Present State 'q0'	Present State 'q1'	Present State 'q2'	Present State 'q3'	Present State 'q4'
0	BRq1	BRq1	ORq2	-	OLq4
1	1Rq2	1Rq2	1Rq2	-	1Lq4
B	BRq1	BLq3	BLq4	OLqf	BRqf

6. Let x and y are two positive integers. Obtain a TM to perform x+y

Solution

STEP 1: Storing of x and y in the tape

STEP 2: Transition diagram

STEP 3: Output

STEP 4: Transition table

Given two positive integers x and y, design a Turing machine that computes $x + y$.

We first have to choose some convention for representing positive integers. For simplicity, we will use unary notation in which any positive integer x is represented by $w(x) \in \{1\}^+$, such that

$$|w(x)| = x$$

that is, for example, number "2" will be represented by " $w(2) = 11$ ", number "3" will be represented by " $w(3) = 111$ ", and number "5" will be represented by " $w(5) = 11111$ ", and so on.

We must also decide how x and y are placed on the tape initially and how their sum is to appear at the end of the computation.

We will assume that $w(x)$ and $w(y)$ are on the tape in this unary notation, separated by a single 0, with the read-write head on the leftmost symbol of $w(x)$.

After the computation, $w(x + y)$ will be on the tape followed by a single 0, and the read-write head will be positioned at the left end of the result. We therefore want to design a Turing machine for performing the computation

$$q_0 w(x) 0 w(y) \rightarrow_M^* q_f (w(x+y) 0,$$

where q_f is a final state.

Constructing a program for this is relatively simple. All we need to do is to move the

separating 0 to the right end of $w(y)$, so that the addition amounts to nothing more than the *coalescing* of the two strings.

To achieve this, we construct $M = (Q, \Sigma, \Gamma, q_0, \perp, F)$, with $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $F = \{q_4\}$, with the delta transitions defined as follow:

$(q_0, 1) \rightarrow (q_0, 1, R)$	$(q_1, 1) \rightarrow (q_1, 1, R)$	$(q_3, 1) \rightarrow (q_3, 1, L)$
$(q_0, 0) \rightarrow (q_1, 1, R)$	$(q_1, \perp) \rightarrow (q_2, \perp, L)$	$(q_3, \perp) \rightarrow (q_4, \perp, R)$
	$(q_2, 1) \rightarrow (q_3, 0, L)$	

Note that in moving the 0 right we temporarily create an extra 1, a fact that is remembered by putting the machine into state q_1 .

The transition $(q_2, 1) \rightarrow (q_3, 0, L)$ is needed to remove this at the end of the computation.

This can be seen from the sequence of instantaneous descriptions for adding 111 (= 3) to 11 (= 2) to yield 11111 (=5):

The transitions (following the table above) are as follows:

$q_0 111011 \rightarrow 1q_0 11011 \rightarrow 11q_0 1011 \rightarrow 111q_0 011 \rightarrow 1111q_1 11 \rightarrow 11111q_1 1 \rightarrow$
 $111111q_1 \perp \rightarrow$
 $\rightarrow 111111q_2 1 \perp \rightarrow 11111q_3 10 \perp \rightarrow 111q_3 110 \rightarrow 11q_3 1110 \rightarrow 1q_3 11110 \rightarrow q_3 111110 \rightarrow$
 $\rightarrow q_3 \perp 111110 \rightarrow \perp q_4 111110$

Unary notation, although cumbersome for practical computations, is very convenient for programming Turing machines. The resulting programs are much shorter and simpler than if we had used another representation, such as binary or decimal.

Adding numbers is one of the fundamental operations of any computer, one that plays a part in the synthesis of more complicated instructions. Other basic operations are copying strings and simple comparisons. These can also be done easily on a Turing machine.

7. Design a Turing machine that copies strings of 1's. More precisely, find a machine that performs the computation

$$q_0 w \rightarrow_M^* q_f w w,$$

for any $w \in \{1\}^+$.

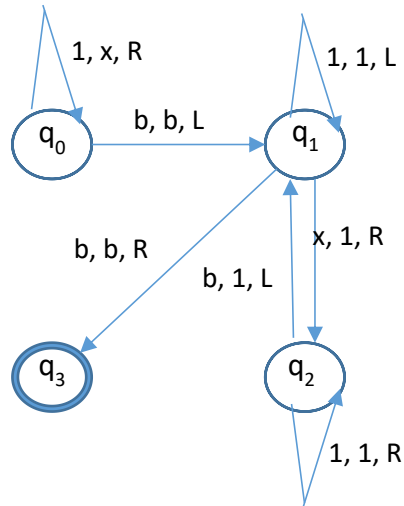
SOLUTION

To solve the problem, we implement the following intuitive process:

1. Replace every 1 by an x
2. Find the rightmost x and replace it with 1.
3. Travel to the right end of the current nonblank region and create a 1 there.
4. Repeat Steps 2 and 3 until there are no more x's.

The solution is shown in the transition graph in DIAGRAM below. It may be a little

hard to see at first that the solution is correct, so let us trace the program with the simple string 11. The computation performed in this case is given below after the figure.



$q_0 11 \rightarrow x q_0 1 \rightarrow x x q_0 \perp \rightarrow x q_1 x \rightarrow$
 $\rightarrow x 1 q_2 \perp \rightarrow x q_1 11 \rightarrow q_1 x 11 \rightarrow$
 $\rightarrow 1 q_2 11 \rightarrow 11 q_2 1 \rightarrow 111 q_2 \perp \rightarrow$
 $\rightarrow 11 q_1 11 \rightarrow 1 q_1 111 \rightarrow q_1 1111 \rightarrow q_1 \perp 1111 \rightarrow q_3 1111$

8. Design a Turing machine that computes the function

$$f(w) = 1 \text{ if } w \text{ is even} \\ = 0 \text{ if } w \text{ is odd:}$$

Solution:

Using the unary representation for w , the Turing machine for the solution of this problem is.

$$\begin{aligned}
 (q_0, 1) &= (q_1, \perp, R); \\
 (q_1, 1) &= (q_0, \perp, R); \\
 (q_0, \perp) &= (q_2, 1, L); \\
 (q_1, \perp) &= (q_2, 0, L); \\
 (q_2, \perp) &= (q_3, \perp, R);
 \end{aligned}$$

with $F = \{q_3\}$.

9.. Design a Turing machine that computes the function

$$\begin{aligned}
 f(x) &= x - 2 && \text{if } x > 2 \\
 &= 0 && \text{if } x \leq 2:
 \end{aligned}$$

Solution:

Using unary representation for x , the Turing machine for the solution of this problem is

$$(q_0, 1) = (q_1, \perp, R);$$

$(q_0, \perp) = (q_4, \perp, L);$
 $(q_1, 1) = (q_2, \perp, R);$
 $(q_1, \perp) = (q_4, \perp, L);$
 $(q_2, 1) = (q_3, 1, L);$
 $(q_2, \perp) = (q_4, \perp, L);$
 $(q_3, \perp) = (q_4, \perp, R);$

with $F = \{q_4\}$.

10. Design Turing machines to compute the following functions for x and y positive integers represented in unary.

(a) $f(x) = 2x + 1$.

(b) $f(x, y) = x + 2y$.

(c) $f(x, y) = 2x + 3y$.

(d) $f(x) = x \bmod 5$.

(e) $f(x) = \lfloor x/2 \rfloor$ where $\lfloor x/2 \rfloor$ denotes the largest integer less than or equal to $x/2$.

Solution:

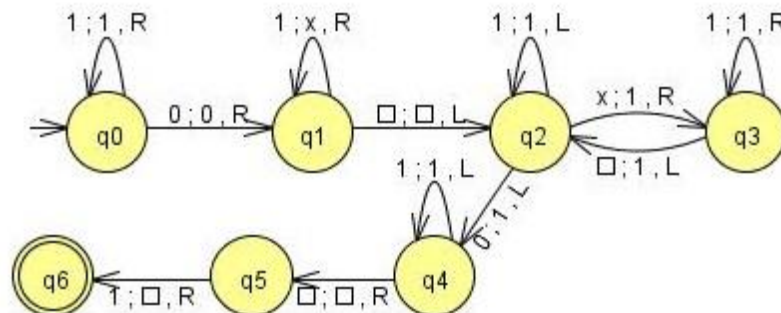
(a) Use the construction in Example 9.10 to duplicate x , then add a symbol 1 to the resulting string. The Turing machine has the following transition functions.

$(q_0, 1) = (q_0, x, R);$
 $(q_0, \perp) = (q_1, \perp, L);$
 $(q_1, 1) = (q_1, 1, L);$
 $(q_1, x) = (q_2, 1, R);$
 $(q_2, 1) = (q_2, 1, R);$
 $(q_2, \perp) = (q_1, 1, L);$
 $(q_1, \perp) = (q_3, 1, L);$
 $(q_3, \perp) = (q_4, \perp, R);$

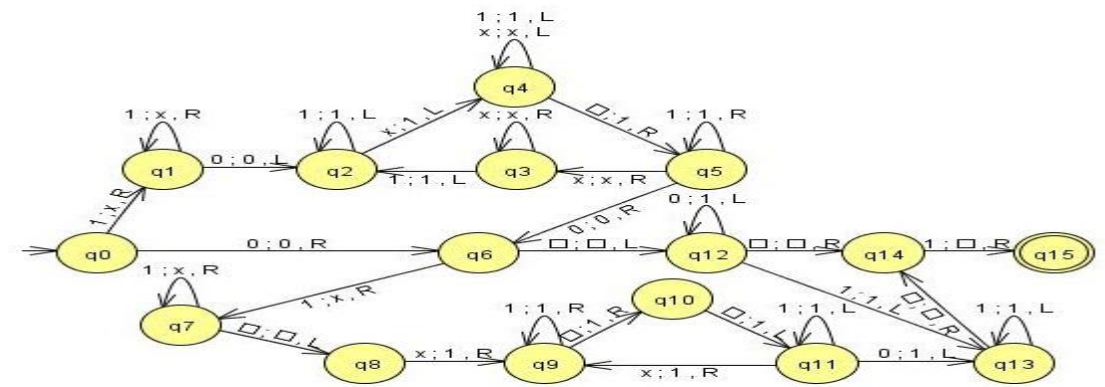
with $F = \{q_4\}$.

(b) For a given problem $w(x)0w(y)$, use the construction in Example 9.10 to duplicate $w(y)$, then add to $w(x)$ to get the resulting string $w(x)w(y)w(y)$.

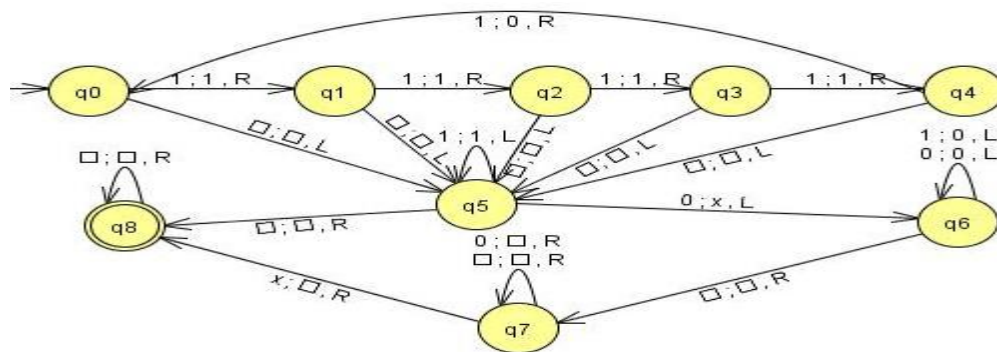
The transition graph is



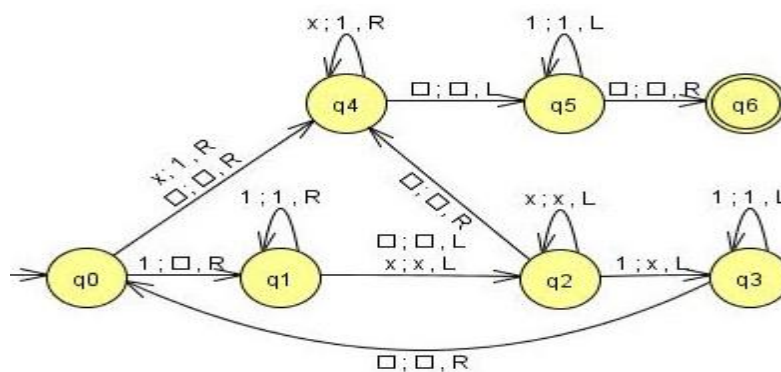
(c) Similar to (a) and (b). Use the construction in Example 9.10 to duplicate x , and triple y , then concatenate $2x$ and $3y$ to obtain the resulting string. The solution is very long as can be seen from its transition graph.



(d) Start from the leftmost 1, the read-write head travels from left to right and replace every 5th 1 by 0 on the move until the rightmost 1 is encountered. Then remove all the 1's on the left of any 0's. The transition graph of a Turing machine implementing the above idea is given below.



(e) Start by removing the leftmost 1, then move to the rightmost 1 and mark it by replacing it with x. The read-write head then moves back to the leftmost 1 and repeat the above process until there are no more 1's. Finally, we replace all the marked x's by 1's to complete the computation. The transition graph of a Turing machine with $F = \{q_6\}$ is given below.



11. Construct a Turing Machine for language $L = \{0^n 1^n 2^n \mid n \geq 1\}$

Solution

The language $L = \{0^n 1^n 2^n \mid n \geq 1\}$ represents a kind of language where we use only 3 character, i.e., 0, 1 and 2. In the beginning language has some number of 0's followed by equal number of 1's and then followed by equal number of 2's. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by \$ sign.

Input : 0 0 1 1 2 2

Output : Accepted

Input : 0 0 0 1 1 1 2 2 2 2

Output : Not accepted

Assumption: We will replace 0 by X, 1 by Y and 2 by Z

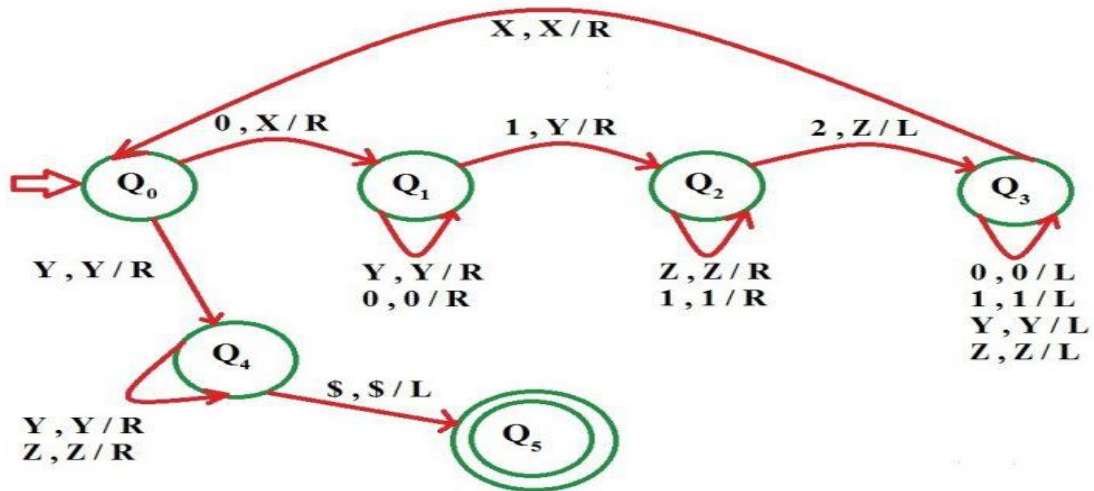
Approach used –

First replace a 0 from front by X, then keep moving right till you find a 1 and replace this 1 by Y. Again, keep moving right till you find a 2, replace it by Z and move left. Now keep moving left till you find a X. When you find it, move a right, then follow the same procedure as above.

A condition comes when you find a X immediately followed by a Y. At this point we keep moving right and keep on checking that all 1's and 2's have been converted to Y and Z. If not then string is not accepted. If we reach \$ then string is accepted.

- **Step-1:**
Replace 0 by X and move right, Go to state Q1.
- **Step-2:**
Replace 0 by 0 and move right, Remain on same state
Replace Y by Y and move right, Remain on same state
Replace 1 by Y and move right, go to state Q2.
- **Step-3:**
Replace 1 by 1 and move right, Remain on same state
Replace Z by Z and move right, Remain on same state
Replace 2 by Z and move right, go to state Q3.
- **Step-4:**
Replace 1 by 1 and move left, Remain on same state
Replace 0 by 0 and move left, Remain on same state
Replace Z by Z and move left, Remain on same state
Replace Y by Y and move left, Remain on same state
Replace X by X and move right, go to state Q0.
- **Step-5:**
If symbol is Y replace it by Y and move right and Go to state Q4
Else go to step 1
- **Step-6:**
Replace Z by Z and move right, Remain on same state

Replace Y by Y and move right, Remain on same state
 If symbol is \$ replace it by \$ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q5



12. Construct a Turing Machine for language $L = \{0^{2n}1^n \mid n \geq 0\}$

Solution

The language $L = \{0^{2n}1^n \mid n \geq 0\}$ represents a kind of language where we use only 2 symbols, i.e., 0 and 1. In the beginning language has some number of 0's followed by exactly half number of 1's. Any such string which falls in this category will be accepted by this language.

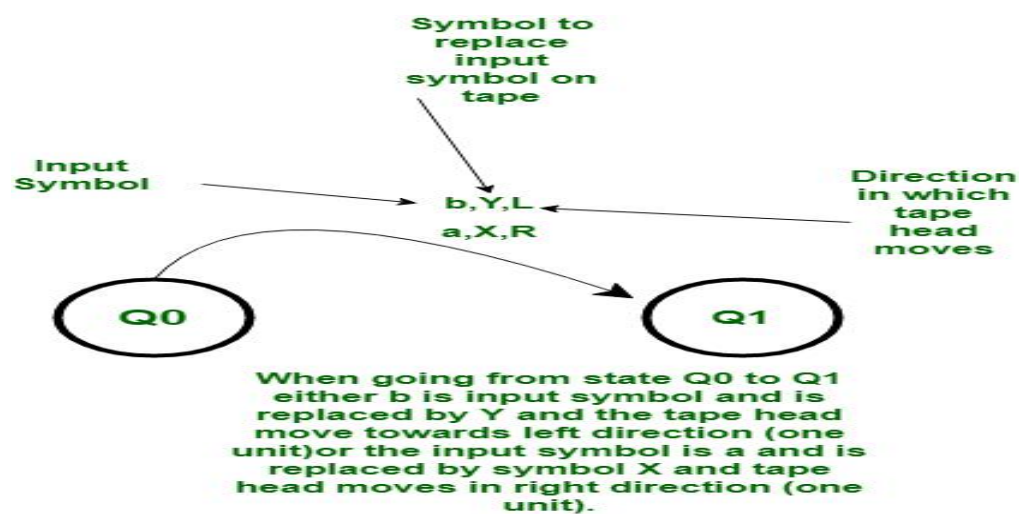
Examples :

Input : 001**Output :** YES

Input : 00001**Output :** NO

Input : ϵ or empty string**Output :** YES

Basic Representation –

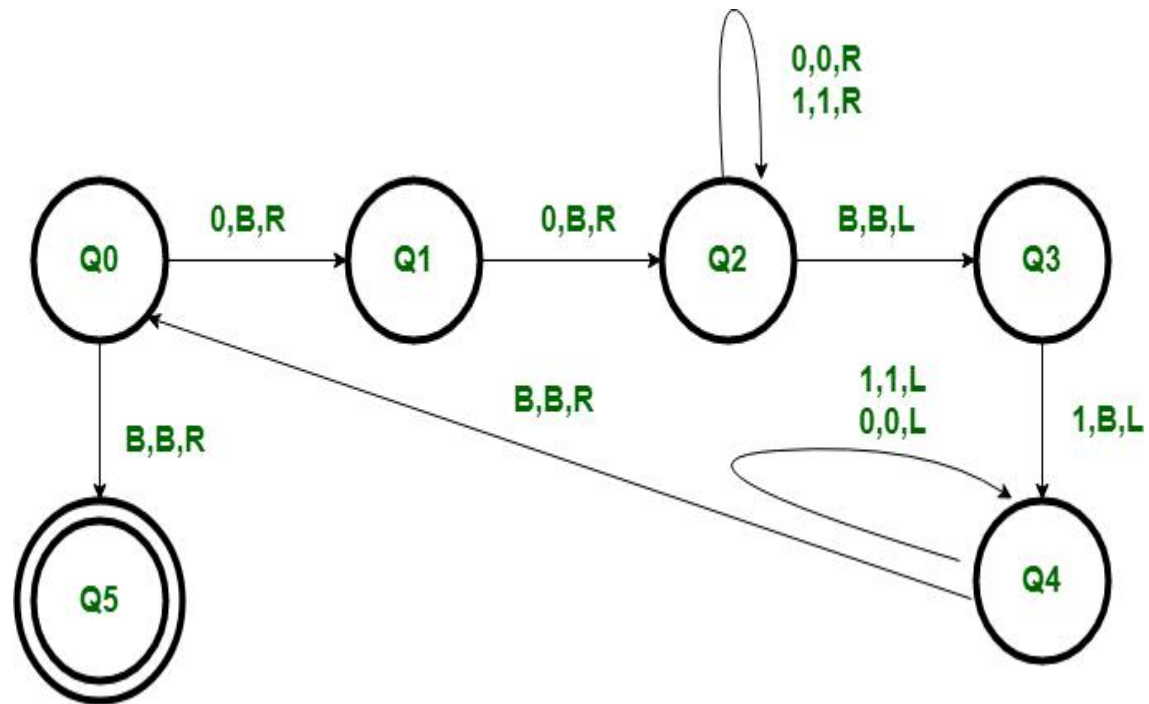


Start of Computation :

The tape contains the input string w , the tape head is on the leftmost symbol of w , and the Turing machine is in the start state Q_0 .

Basic Idea :

The tape head reads the leftmost symbol of w , which is 0 and makes it blank then the next left most 0 is made blank after this we traverse to the rightmost 1 of the string and make it blank. In nth way we have reduced the string to $0^{2n-2}1^{n-1}$. If the string belongs to language L then at end empty string will be left and hence gets accepted by the machine.

**Meanings of symbols used:**

R, L – direction of movement of one unit on either side.

B-Blank.

0, 1 – symbols whose combination string is to be tested.

Working Procedure :

- **Step-1:**
As we know the string will have twice number of zeros in the starting than the number of ones. So, we will first make the first two zeros Blank and go from state Q_0 to Q_1 and from state Q_1 to Q_2 .
- **Step-2:**
After making them Blank we will traverse to the end of the string till we get the rightmost 1 in state Q_3 and make it Blank reaching state Q_4 .
- **Step-3:**
Now we will traverse back till we get the left most zero in the string and return to the state Q_0 from Q_4 .
- **Step-4:**
We are just reducing the string by making left most two zeros Blank and rightmost 1 Blank and if the string belongs to the language then it will be left

empty and hence get accepted at state Q5 which is Final state. If the string is empty it will also get accepted at Q5.

13. Construct a Turing Machine for language $L = \{ww^r \mid w \in \{0, 1\}^*\}$

Solution

The language $L = \{ww^r \mid w \in \{0, 1\}^*\}$ represents a kind of language where you use only 2 character, i.e., 0 and 1. The first part of language can be any string of 0 and 1. The second part is the reverse of the first part. Combining both these parts out string will be formed. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by \$ sign.

For example, if first part $w = 11001$ then second part $w^r = 10011$. It is clearly visible that w^r is the reverse of w , so the string 1100110011 is a part of given language.

Examples –

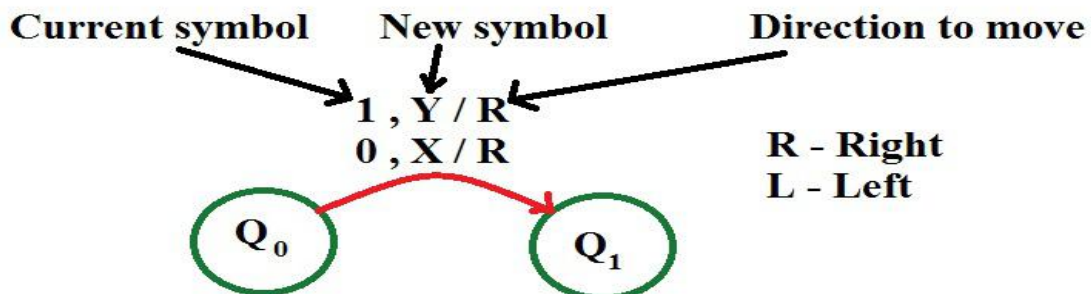
Input : 00111100

Output : Accepted

Input : 10100101

Output : Accepted

Basic Representation –



**If current symbol is 0, replace it by X and move right
OR**

If current symbol is 1, replace it by Y and move right

Assumption: We will replace 0 by Y and 1 by X.

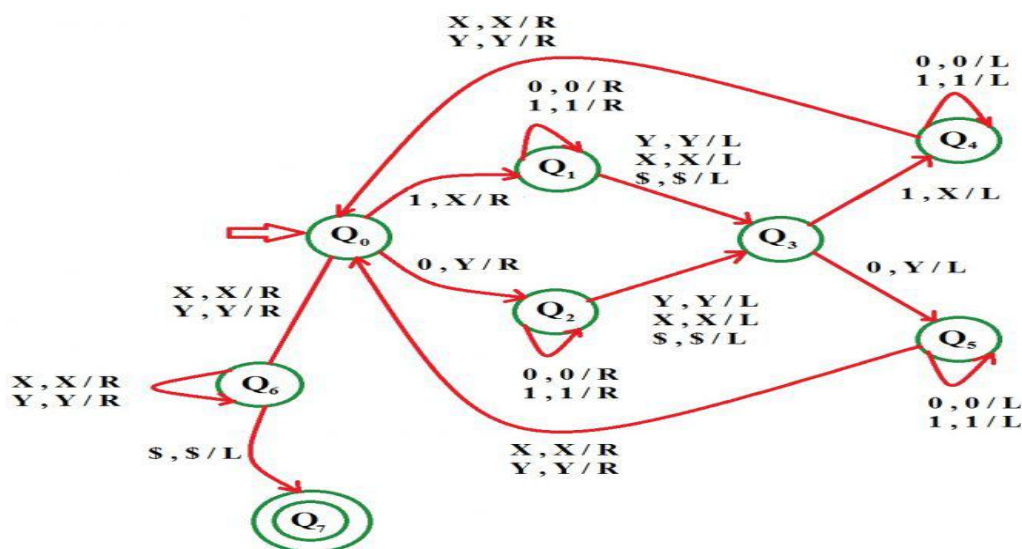
Approach Used –

First check the first symbol, if it's 0 then replace it by Y and by X if it's 1. Then go to the end of string. So last symbol is same as first. We replace it also by X or Y depending on it.

Now again come back to the position next to the symbol replace from the starting and repeat the same process as told above.

One important thing to note is that since w^r is reverse of w of both of them will have equal number of symbols. Every time replace a n th symbol from beginning of string, replace a corresponding n th symbol from the end.

- **Step-1:**
If symbol is 0 replace it by Y and move right, Go to state Q2
If symbol is 1 replace it by X and move right, Go to state Q1
 - **Step-2:**
If symbol is 0 replace it by 0 and move right, remain on same state
If symbol is 1 replace it by 1 and move right, remain on same state
-
- If symbol is X replace it by X and move right, Go to state Q3
If symbol is Y replace it by Y and move right, Go to state Q3
If symbol is \$ replace it by \$ and move right, Go to state Q3
- **Step-3:**
If symbol is 1 replace it by X and move left, Go to state Q4
If symbol is 0 replace it by Y and move left, Go to state Q5
 - **Step-4:**
If symbol is 1 replace it by 1 and move left
If symbol is 0 replace it by 0 and move left
Remain on same state
 - **Step-5:**
If symbol is X replace it by X and move right
If symbol is Y replace it by Y and move right
Go to state Q0
 - **Step-6:**
If symbol is X replace it by X and move right
If symbol is Y replace it by Y and move right
Go to state Q6
ELSE
Go to step 1
 - **Step-7:**
If symbol is X replace it by X and move right, Remain on same state
If symbol is Y replace it by Y and move right, Remain on same state
If symbol is \$ replace it by \$ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q7



13. Construct a Turing machine for $L = \{a^i b^j c^k \mid i < j < k; i \geq 1\}$

Solution

In given language $L = \{a^i b^j c^k \mid i < j < k; i \geq 1\}$, every string of 'a', 'b' and 'c' have certain number of a's, then certain number of b's and then certain number of c's. The condition is that count of 1st symbols should be atleast 1. 'b' and 'c' can have thereafter be as many but count of a is less than count of 'b' and count of 'b' is less than count of 'c'. Assume that string is ending with '\$'.

Examples:

Input: a a a b b c

Here $a = 3, b = 2, c = 1$ but $|a| < |b| < |c|$

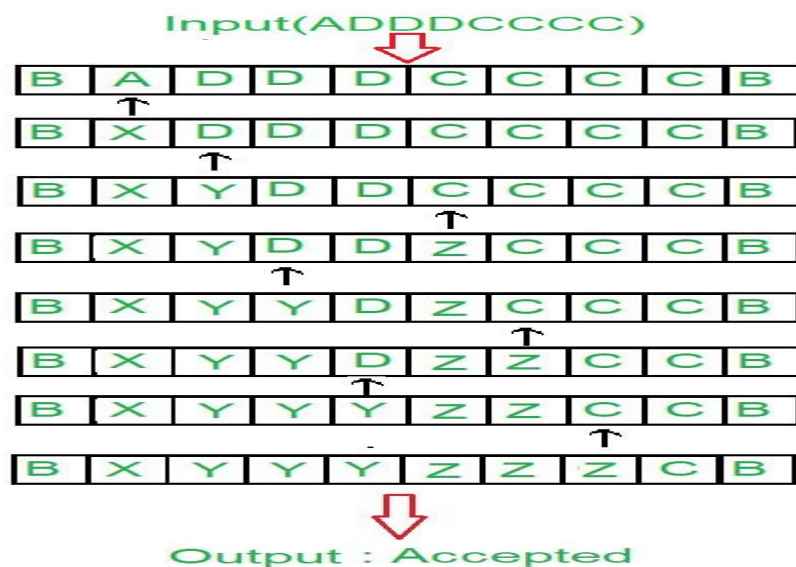
Output: NOT ACCEPTED

Input: a b b c c c

Here $a = 1, b = 2, c = 3$

Output: ACCEPTED

Tape Representation:



Approach:

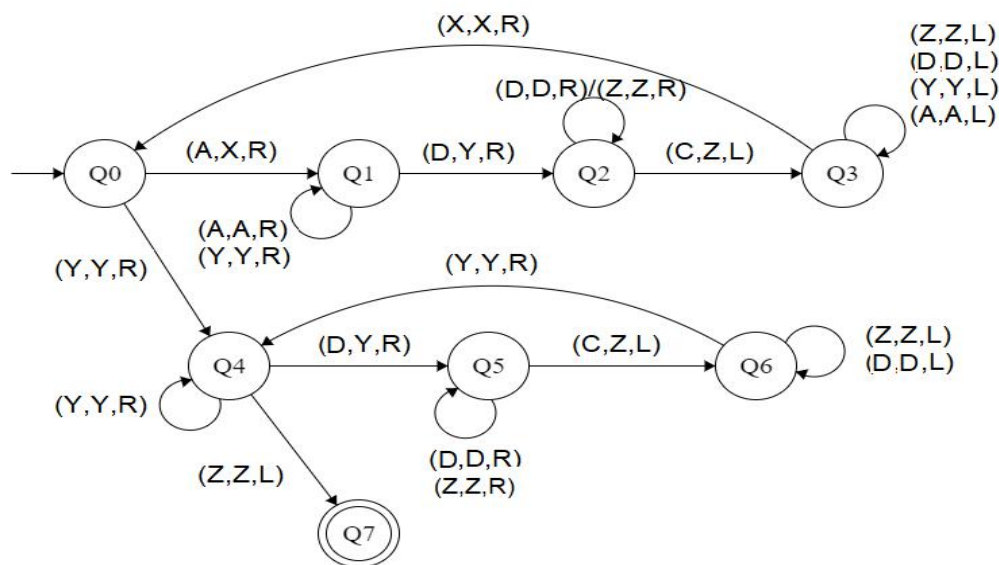
1. Comparing two elements by making D & C as a single element.
2. After that Comparing D & C.
3. If $|A|$ is greater than $|(D, C)|$, then it is not accepted.
4. If $|C|$ is greater than $|D|$, then it is not accepted.
5. Else it is accepted.

Steps:

- **Step-1:** Convert A into X and move right and goto step 2.If Y is found ignore it and move right to step-5.

- **Step-2:** Keep ignoring A and Y and move towards right. Convert D into Y and move right and goto step-3.
- **Step-3:** Keep ignoring D and Z and move towards right.If C is found make it Z and move left to step 4.
- **Step-4:** Keep ignoring Z, A, Y and D and move towards left.If X is found ignore it and move right and goto step-1.
- **Step-5:** Keep ignoring Y and move towards right. Ignore Z move left and goto step-8.If D is found make it Y and move right to step-6.
- **Step-6:** Keep ignoring D and Z and move towards right.Convert C into Z and move left and goto step-7.
- **Step-7:** Keep ignoring D and Z and move towards left.If Y is found ignore it and move right and goto step-5.
- **Step-8:** Stop the Machine (String is accepted)

State transition diagram :



Here, **Q0** shows the initial state and **Q1, Q2, Q3, Q4, Q5, Q6** shows the transition state and **Q7** shows the final state. A, C, D are the variables used and R, L shows right and left.

Explanation:

- Using Q0, when A is found make it X and go to the right and to state Q1. And, when Y is found ignore it and go to the right and to state Q4
- On the state Q1, ignore all A and Y and goto right. If D found, make it Y and goto right into next state Q2.
- In Q2, ignore all D, Z and move right. If C found, make it Z move left and to Q3.
- In Q3 state, ignore all Z, D, Y, A and move left. If X found, ignore it move right to Q0.
- In Q4, ignore all Y and move right. If Z found, ignore it move left to state Q6. If D is found, make it Y and move to the right to Q5.
- In Q5 state, ignore all D, Z and move right. If C found, make it Z move left to state Q6

- In Q6, ignore all D, Z and move left. If Y found, ignore it and move right to state Q4.
- If Q7 state is reached it will produce the result of the acceptance of string.

14. Construct a Turing Machine for language $L = \{ww \mid w \in \{0,1\}^*\}$

Solution

The language $L = \{ww \mid w \in \{0, 1\}^*\}$ tells that every string of 0's and 1's which is followed by itself falls under this language. The logic for solving this problem can be divided into 2 parts:

1. Finding the mid point of the string
2. After we have found the mid point we match the symbols

Example – Lets understand it with the help of an example. Lets string 1 0 1 1 0 1, so $w = 1 0 1$ and string is of form (ww).

The first thing that we do is to find the midpoint. For this, we convert 1 in the beginning into Y and move right till the end of the string. Here we convert 1 into y.

Now our string would look like Y 0 1 1 0 Y. Now move left till, find a X or Y. When we do so, convert the 0 or 1 right of it to X or Y respectively and then do the same on the right end. Now our string would look like Y X 1 1 X Y. Thereafter, convert these 1's also and finally it would look like Y X Y Y X Y.

At this point, you have achieved the first objective which was to find the midpoint. Now convert all X and Y on the left of midpoint into 0 and 1 respectively, so string becomes 1 0 1 Y X Y. Now, convert the 1 into Y and move right till, find Y in the beginning of the right part of the string and convert this Y into a blank (denoted by B). Now string looks like Y 0 1 B X Y.

Similarly, apply this on 0 and x followed by 1 and Y. After this string looks like Y X Y B B B. Now that you have no 0 and 1 and all X and Y on the right part of the string are converted into blanks so our string will be accepted.

Assumption: We will replace 0 by X and 1 by Y.

Approach Used –

The first thing is to find the midpoint of the string, convert a 0 or 1 from the beginning of the string into X or Y respectively and a corresponding 0 or 1 into X or Y from the end of the string. After continuously doing it a point is reached when all 0's and 1's have been converted into X and Y respectively. At this point, you are on the midpoint of the string. So, our first objective is fulfilled.

Now, convert all X's and Y's on the left of the midpoint into 0's and 1's. At this point the first half the string is in the form of 0 and 1. The second half of the string is in the form of X and Y.

Now, start from the beginning of the string. If you have a 0 then convert it into X and move right till reaching the second half, here if we find X then convert it into a blank(B). Then traverse back till find an X or a Y. We convert the 0 or 1 on the right of it into X or Y respectively and correspondingly, convert its X or Y in the second half of string into a blank(B).

Keep doing this till converted all symbols on the left part of the string into X and Y and all symbols on the right of string into blanks. If any one part is completely converted but still some symbols in the other half are left unchanged then the string will not be accepted. If you did not find an X or Y in the second half for a corresponding 0 or 1 respectively in the first half. Then also string will not be accepted.

Examples:

Input : 1 1 0 0 1 1 0 0

Output : Accepted

Input : 1 0 1 1 1 0 1

Output : Not accepted

- **Step-1:**

If the symbol is 0 replace it by X and move right

If the symbol is 1 replace it by Y and move right,

Go to state Q1 and step 2

If the symbol is X replace it by X and move left or

If the symbol is Y replace it by Y and move left,

Go to state Q4 and step 5

- **Step-2:**

If the symbol is 0 replace it by 0 and move right, remain on the same state

If the symbol is 1 replace it by 1 and move right, remain on the same state

If the symbol is X replace it by X and move left or

If the symbol is Y replace it by Y and move left or

If the symbol is \$ replace it by \$ and move left, Go to state Q2 and step 3

- **Step-3:**

If symbol is 0 replace it by X and move left, or

If symbol is 1 replace it by Y and move left,

Go to state Q3 and step 4

- **Step-4:**

If the symbol is 0 replace it by 0 and move left, remain on the same state

If the symbol is 1 replace it by 1 and move left, remain on the same state

If the symbol is X replace it by X and move right or

If the symbol is Y replace it by Y and move right,

Go to state Q0 and step 1

- **Step-5:**

If symbol is X replace it by X and move left or

If symbol is Y replace it by Y and move left

Go to state Q4 and step 6

- **Step-6:**

If symbol is X replace it by 0 and move left, remain on same state

If symbol is Y replace it by 1 and move left, remain on same state

If symbol is \$ replace it by \$ and move right
Go to state Q4 and step 7

- **Step-7:**

If symbol is 0 replace it by X and move right, go to state Q6 and step 8
If symbol is 1 replace it by Y and move right, go to state Q7 and step 9

If symbol is B replace it by B and move left, STRING ACCEPTED, GO TO FINAL STATE Q9

- **Step-8:**

If symbol is 0 replace it by 0 and move right, remain on same state
If symbol is 1 replace it by 1 and move right, remain on same state
If symbol is B replace it by B and move right, remain on same state

If symbol is X replace it by B and move left
Go to state Q8 and step 10

- **Step-9:**

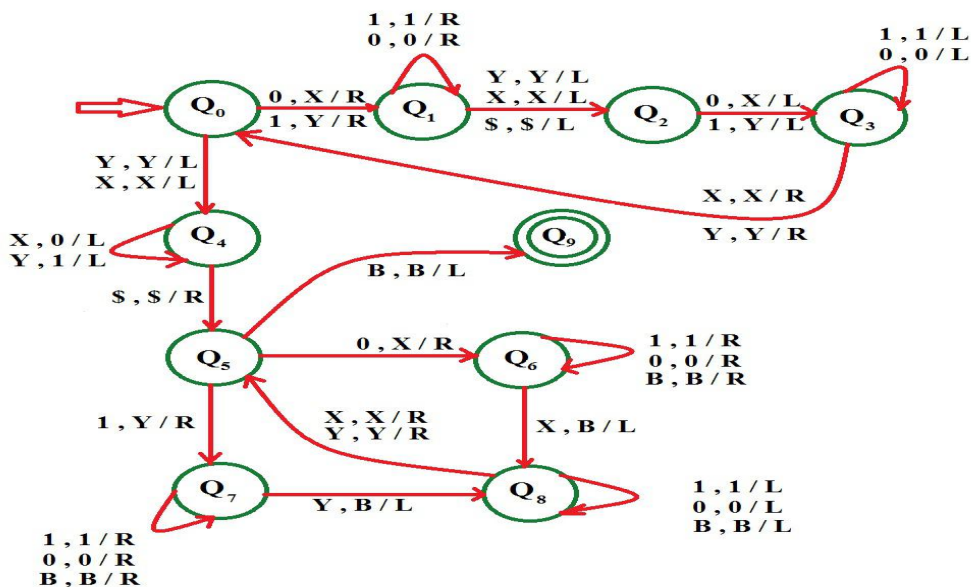
If symbol is 0 replace it by 0 and move right, remain on same state
If symbol is 1 replace it by 1 and move right, remain on same state
If symbol is B replace it by B and move right, remain on same state

If symbol is Y replace it by B and move left
Go to state Q8 and step 10

-
- **Step-10:**

If symbol is 0 replace it by 0 and move left, remain on same state
If symbol is 1 replace it by 1 and move left, remain on same state
If symbol is B replace it by B and move left, remain on same state

If symbol is Y replace it by Y and move right or
If symbol is X replace it by X and move right
Go to state Q5 and step 7



15. Construct Turing machine for $L = \{a^n b^m a^{(n+m)} \mid n, m \geq 1\}$

Solution

$L = \{a^n b^m a^{(n+m)} \mid n, m \geq 1\}$ represents a kind of language where we use only 2 character, i.e., a and b. The first part of language can be any number of "a" (at least 1). The second part be any number of "b" (at least 1). The third part of language is a number of "a" whose count is sum of count of a's in first part of string and count of b's in second part of string. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by \$ sign.

Examples:

Input : a a b b b a a a a // n=2, m=3

Output : Accepted

Input : a a b a a a // n=2, m=1

Output : Not accepted

Approach used –

1. Convert "a" in first part into "X" and then move right ignoring all intermediate symbols. When "a" is encountered just after "b" then convert it into "Z" and move left and stop at the position just next to "X". Repeat the above procedure.
2. When all a's in first part have been converted then apply the same process on second part. Convert "b" into "Y" and "a" into "Z" in the third part.

When entire first and second part has been converted and if third part is also converted then string will be accepted else not.

Steps –

Step-0: Convert "a" into "X", move right and go to state 1. If symbol is "b", ignore it, move right and go to state-4.

Step-1: If symbol is "a", ignore it and move right, remain on same state. If symbol is "b", ignore it, move right and go to state-2.

Step-2: If symbol is "Z", ignore it and move right, remain on same state. If symbol is "b", ignore it and move right, remain on same state and if symbol is "a", convert it into "Z", move left and go to state-3.

Step-3: If symbol is "Z", ignore it and move left, remain on same state. If symbol is "b", ignore it and move left, remain on same state. If symbol is "a", ignore it and move left, remain on same state, and if symbol is "X", ignore it and move right, go to state-0.

Step-4: If symbol is "b", ignore it and move left, go to state 5, and if symbol is "Z", ignore it and move left, go to state-5.

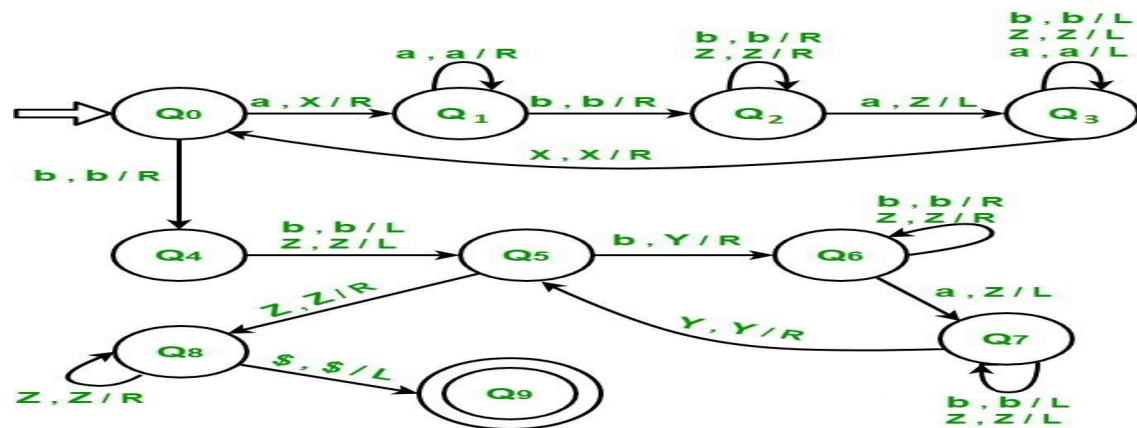
Step-5: Convert "b" into "Y", move right and go to state 6, and if symbol is "Z", ignore it and move right, go to state-8.

Step-6: If symbol is “Z”, ignore it and move right, remain on same state. If symbol is “b”, ignore it and move right, remain on same state, and if symbol is “a”, convert it into “Z”, move left and go to state-7.

Step-7: If symbol is “Z”, ignore it and move left, remain on same state. If symbol is “b”, ignore it and move left, remain on same state, and if symbol is “Y”, ignore it and move right, go to state-5.

Step-8: If symbol is “Z”, ignore it and move right, remain on same state, and if symbol is “\$”, ignore it and move left, go to state-9.

Step-9: String ACCEPTED



16. Construct a Turing Machine for language $L = \{0^n 1^n 2^n \mid n \geq 1\}$

Solution

The language $L = \{0^n 1^n 2^n \mid n \geq 1\}$ represents a kind of language where we use only 3 character, i.e., 0, 1 and 2. In the beginning language has some number of 0's followed by equal number of 1's and then followed by equal number of 2's. Any such string which falls in this category will be accepted by this language. The beginning and end of string is marked by \$ sign.

Examples –

Input : 0 0 1 1 2 2

Output : Accepted

Input : 0 0 0 1 1 1 2 2 2 2

Output : Not accepted

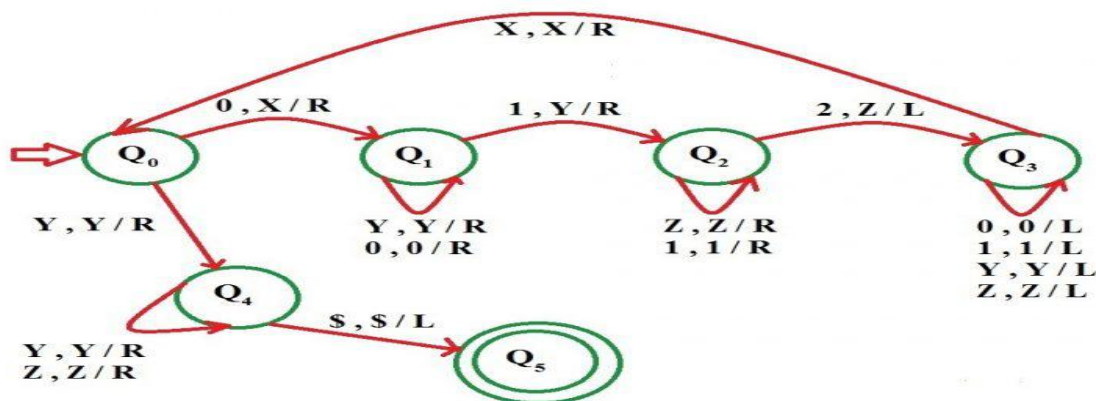
Assumption: We will replace 0 by X, 1 by Y and 2 by Z

Approach used –

First replace a 0 from front by X, then keep moving right till you find a 1 and replace this 1 by Y. Again, keep moving right till you find a 2, replace it by Z and move left. Now keep moving left till you find a X. When you find it, move a right, then follow the same procedure as above.

A condition comes when you find a X immediately followed by a Y. At this point we keep moving right and keep on checking that all 1's and 2's have been converted to Y and Z. If not then string is not accepted. If we reach \$ then string is accepted.

- **Step-1:**
Replace 0 by X and move right, Go to state Q1.
- **Step-2:**
Replace 0 by 0 and move right, Remain on same state
Replace Y by Y and move right, Remain on same state
Replace 1 by Y and move right, go to state Q2.
- **Step-3:**
Replace 1 by 1 and move right, Remain on same state
Replace Z by Z and move right, Remain on same state
Replace 2 by Z and move right, go to state Q3.
- **Step-4:**
Replace 1 by 1 and move left, Remain on same state
Replace 0 by 0 and move left, Remain on same state
Replace Z by Z and move left, Remain on same state
Replace Y by Y and move left, Remain on same state
Replace X by X and move right, go to state Q0.
- **Step-5:**
If symbol is Y replace it by Y and move right and Go to state Q4
Else go to step 1
- **Step-6:**
Replace Z by Z and move right, Remain on same state
Replace Y by Y and move right, Remain on same state
If symbol is \$ replace it by \$ and move left, STRING IS ACCEPTED, GO TO FINAL STATE Q5



17. Construct a Turing Machine for language $L = \{a^n b^m c^{nm} \mid n \geq 0 \text{ and } m \geq 0\}$

Solution

The language $L = \{a^n b^m c^{nm} \mid n \geq 0 \text{ and } m \geq 0\}$ represents a kind of language where we use only 3 symbols, i.e., a, b and c. In the beginning language has n number of a's

followed by m number of b's and then n*m number of c's. Any such string which falls in this category will be accepted by this language.

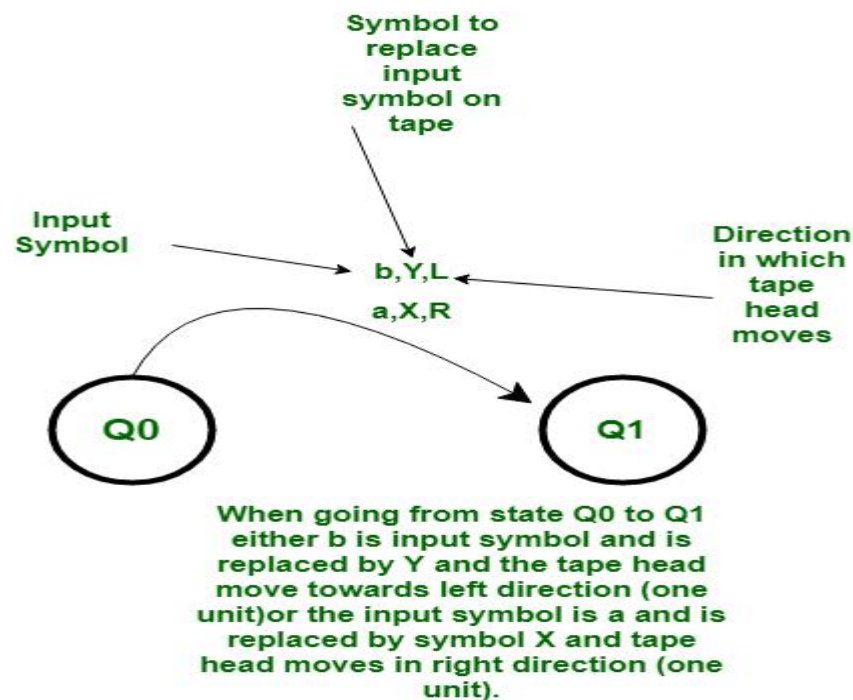
Examples:

Input : abbcc**Output :** YES

Input : abbbcc**Output :** NO

Input : ϵ or empty string**Output :** YES

Basic Representation :

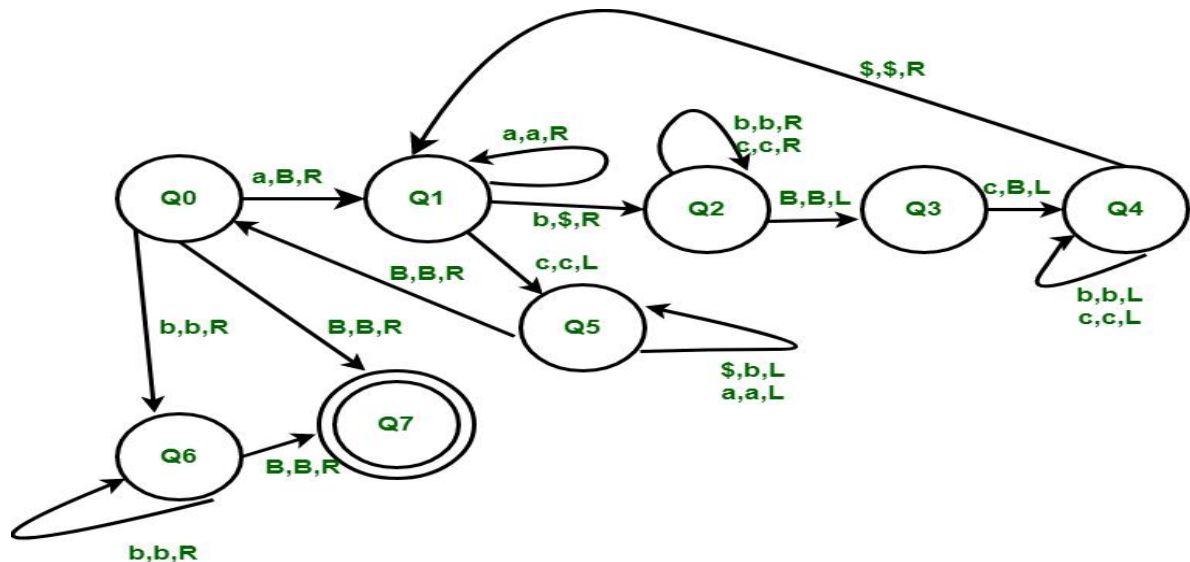


Start of Computation :

The tape contains the input string w, the tape head is on the leftmost symbol of w, and the Turing machine is in the start state Q0.

Basic Idea :

The tape head reads the leftmost symbol of w, which is a and at start only we will make it Blank. Then we will traverse to make leftmost b a \$ and replace rightmost c by a Blank, we will do this zigzag pattern of replacing b by \$ and c by Blank till all b are not replaced by \$. After this we will traverse back in left direction till we get leftmost a and replacing all \$ by b in the traversal. After this we have reduced our string into form $a^{n-1}b^m c^{n-m}$ so we can figure if all a's are replaced by blank and if the string belongs to Language L then there will be no c's left hence it will get accepted.



Meanings of symbols used:

R, L – direction of movement of one unit on either side.

B-Blank,

a, b, c -symbols whose combination string is to be tested.

\$-Temporarily symbol to replace b.

Working Procedure :

- **Step-1:**
We first replace leftmost a by Blank and then traverse to replace leftmost b by \$ and rightmost c by Blank. Repeat this step from state Q1 till there is no more b left.
- **Step-2:**
After replacing all b by \$ we have also replaced m rightmost c's with Blanks and then we will traverse back to left most a and replace all \$ by b's. After this step if check the string it is now reduced to an-1bmcnm-m form. Now we will repeat from step 1 till all a's are not made Blank.
- **Step-3:**
So after all a's are made blank and if the string belonged to Language L then 0 c's must be left which we check at state Q0 and Q6 as only b's will be left and after which if Blank is found then all c's must have been replaced by Blank as we were making c's Blank from rightmost end of the string.
- **Step-4:**
So if we get a Blank symbol at state Q6 the the string is accepted at final state Q7. Also if the string was empty then it will also be accepted as Blank symbol input at state Q0 then it will go to state Q7 and gets accepted.

18. Draw a Turing machine to find 1's complement of a binary number.

Solution

1's complement of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.

Example:

Input \Rightarrow

B	1	0	1	1	1	1	1	0	B
---	---	---	---	---	---	---	---	---	---

Output \Rightarrow

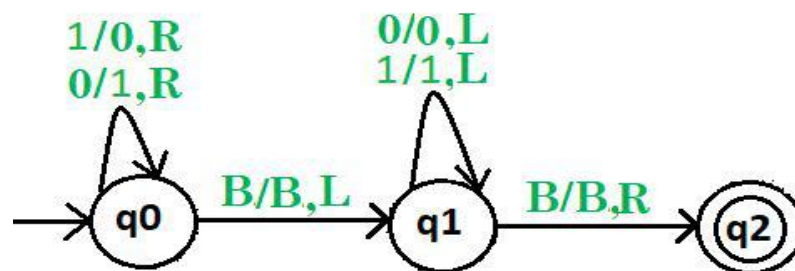
B	0	1	0	0	0	0	0	1	B
---	---	---	---	---	---	---	---	---	---

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.

Steps:

- **Step-1.** Convert all 0's into 1's and all 1's into 0's and go right if B found go to left.
- **Step-2.** Then ignore 0's and 1's and go left & if B found go to right
- **Step-3.** Stop the machine.



Here, **q0** shows the initial state and **q1** shows the transition state and **q2** shows the final state.

And 0, 1 are the variables used and R, L shows right and left.

Explanation:

- State q0 replace '1' with '0' and '0' with '1' and move to right.
- When BLANK is reached move towards left.
- Using state 'q2' we reach start of the string.
- When BLANK is reached move towards right and reaches the final state q2.

19. Draw a Turing machine to find 2's complement of a binary number.

Solution

2's complement of a binary number is 1 added to the 1's complement of the binary number.

Example:

Input \Rightarrow

B	1	0	1	1	1	1	1	0	B
---	---	---	---	---	---	---	---	---	---

Output \Rightarrow

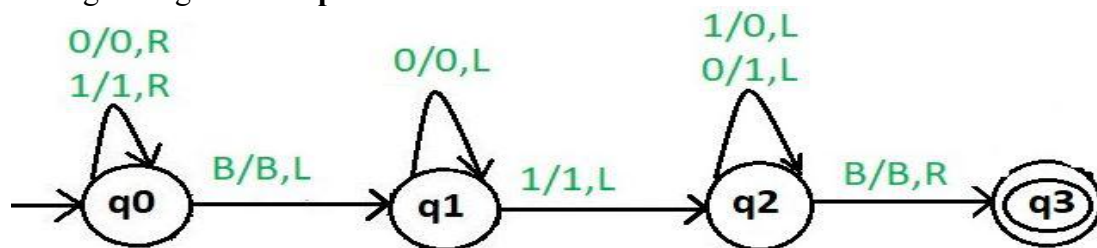
B	0	1	0	0	0	0	1	0	B
---	---	---	---	---	---	---	---	---	---

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.

Steps:

- **Step-1.** First ignore all 0's and 1's and go to right & then if B found go to left.
- **Step-2.** Then ignore all 0's and go left, if 1 found go to left.
- **Step-3.** Convert all 0's into 1's and all 1's into 0's and go to left & if B found go to right and **stop the machine.**



Here, **q0** shows the initial state and **q1 and q2** shows the transition state and **q3** shows the final state.

And 0, 1 are the variables used and R, L shows right and left.

Explanation:

- Using state 'q0' we reach end of the string.
- When BLANK is reached move towards left.
- Using state 'q1' we pass all 0's and move left first 1 is found.
- Pass single '1' and move left.
- Using state 'q2' we complement the each digit and move left.
- When BLANK is reached move towards right and reaches the final state q2.

20. Construct a TM machine for checking the palindrome of the string of even length.

Solution:

Firstly we read the first symbol from the left and then we compare it with the first symbol from right to check whether it is the same.

Again we compare the second symbol from left with the second symbol from right. We repeat this process for all the symbols. If we found any symbol not matching, we cannot lead the machine to HALT state.

Suppose the string is ababbaba Δ . The simulation for ababbaba Δ can be shown as follows:

a	b	a	b	b	a	b	a	Δ	-----
---	---	---	---	---	---	---	---	----------	-------

Now, we will see how this Turing machine will work for ababbaba Δ . Initially, state is q_0 and head points to a as:

a	b	a	b	b	a	b	a	Δ
↑								

We will mark it by * and move to right end in search of a as:

*	b	a	b	b	a	b	a	Δ
		↑						

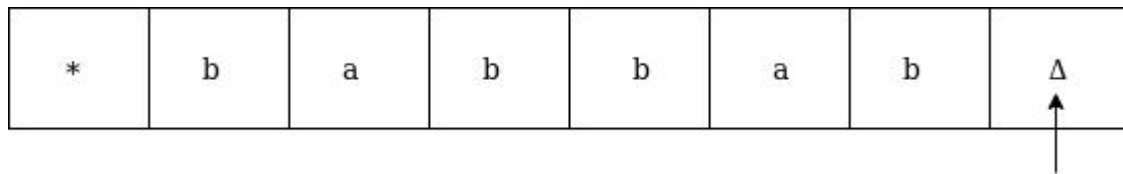
We will move right up to Δ as:

*	b	a	b	b	a	b	a	Δ
								↑

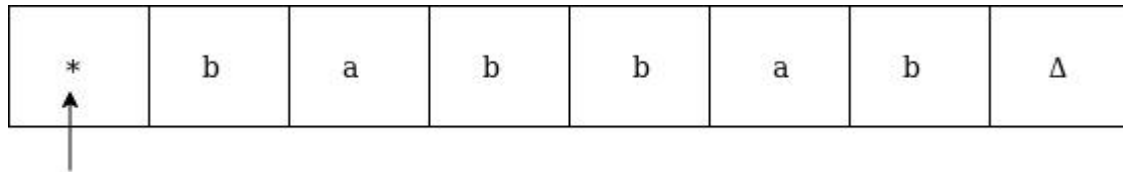
We will move left and check if it is a:

*	b	a	b	b	a	b	a	Δ
							↑	

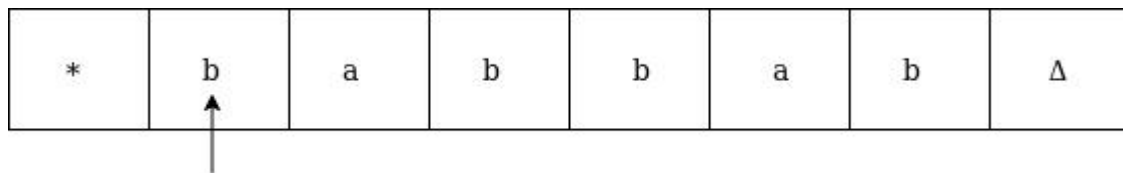
It is 'a' so replace it by Δ and move left as:



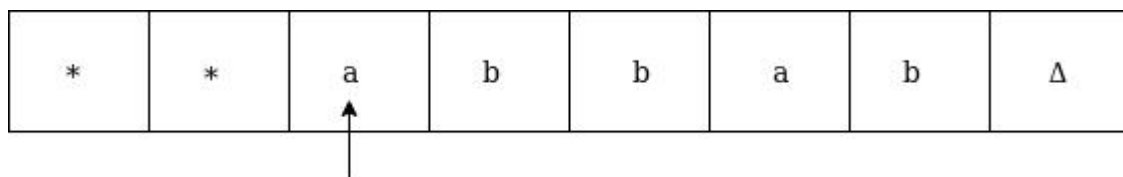
Now move to left up to * as:



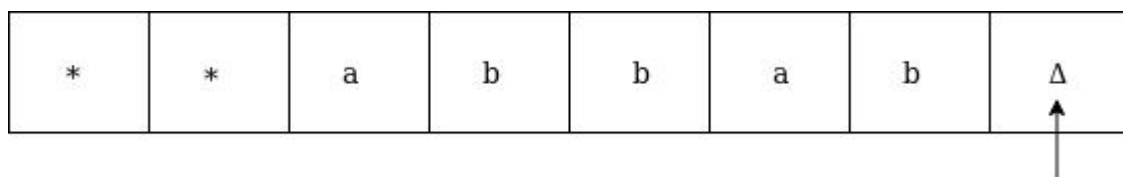
Move right and read it



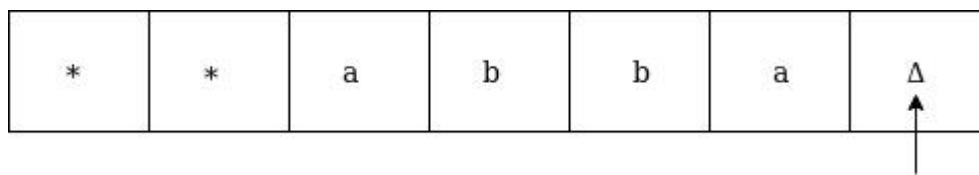
Now convert b by * and move right as:



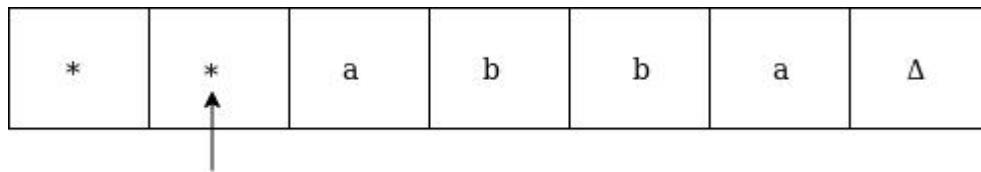
Move right up to Δ in search of b as:



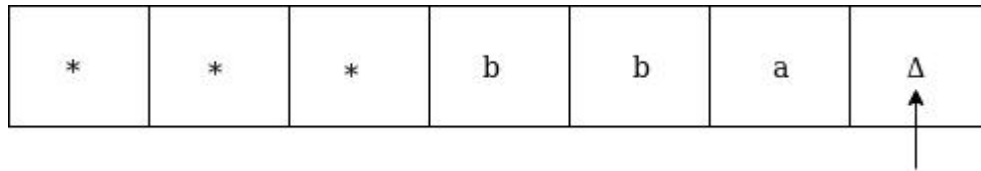
Move left, if the symbol is b then convert it into Δ as:



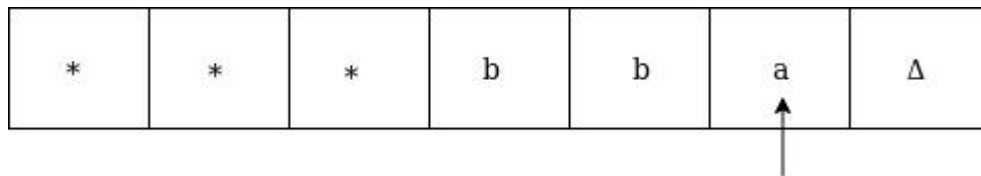
Now move left until * as:



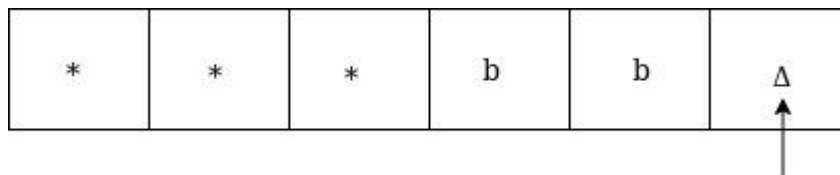
Replace a by * and move right up to Δ as:



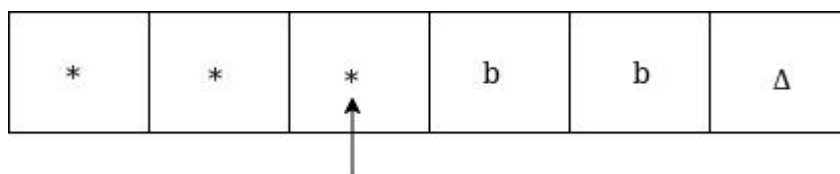
We will move left and check if it is a, then replace it by Δ as:



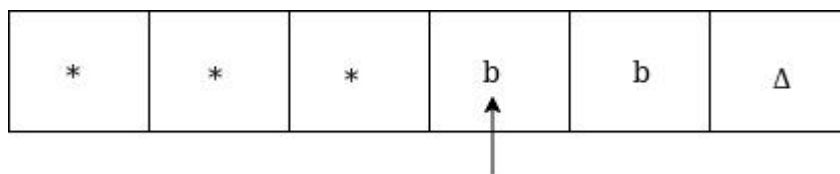
It is 'a' so replace it by Δ as:



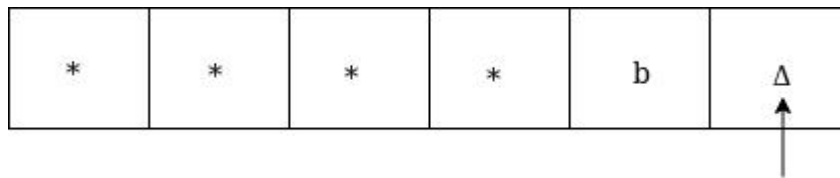
Now move left until *



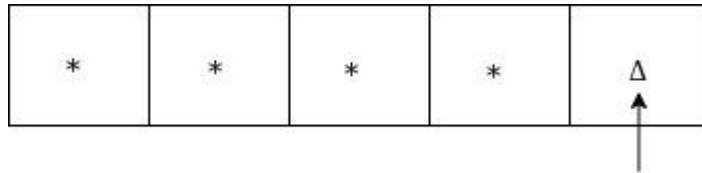
Now move right as:



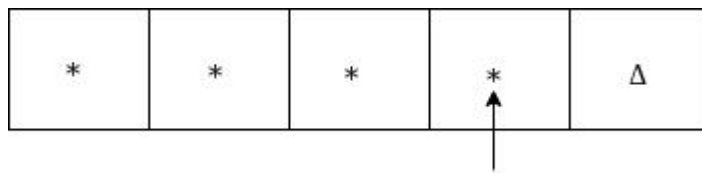
Replace b by * and move right up to Δ as:



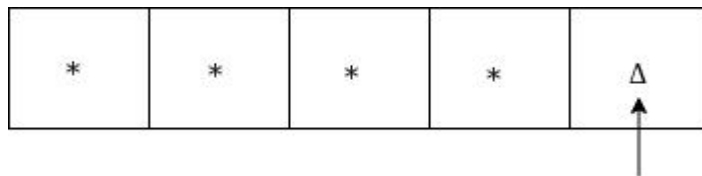
Move left, if the left symbol is b, replace it by Δ as:



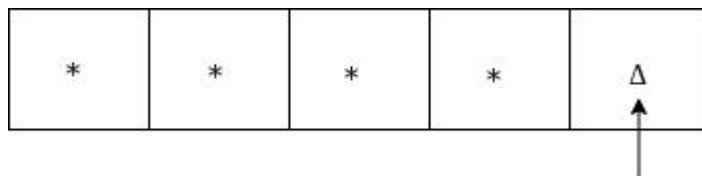
Move left till *



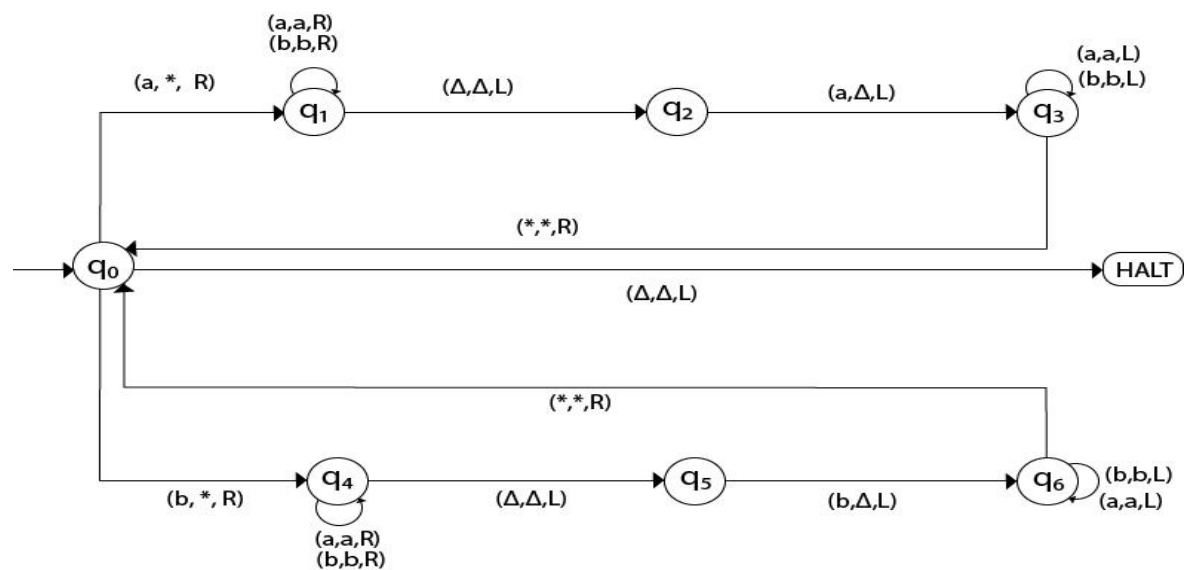
Move right and check whether it is Δ



Go to HALT state



The same TM can be represented by Transition Diagram:



21. Construct a TM machine for checking the palindrome of the string of odd length.

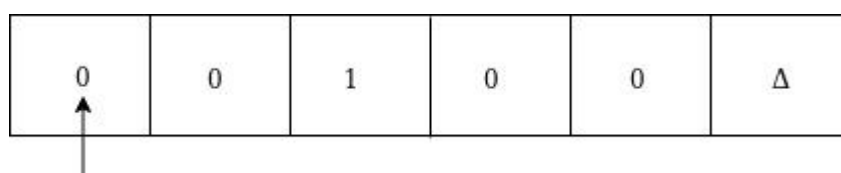
Solution:

Firstly we read the first symbol from left and then we compare it with the first symbol from right to check whether it is the same.

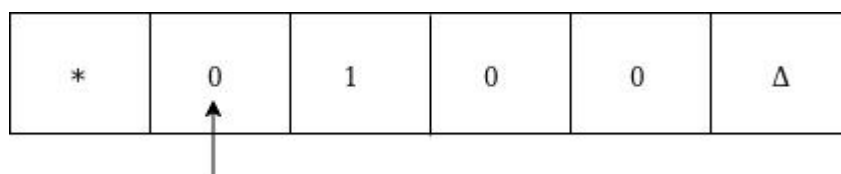
Again we compare the second symbol from left with the second symbol from right. We repeat this process for all the symbols. If we found any symbol not matching, we lead the machine to HALT state.

Suppose the string is 00100Δ. The simulation for 00100Δ can be shown as follows:

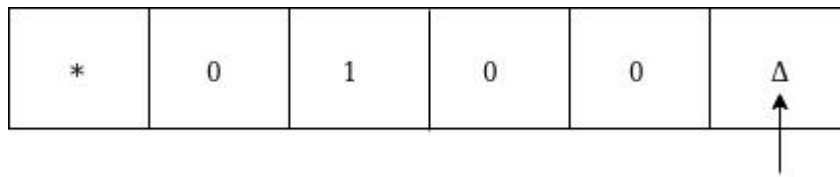
Now, we will see how this Turing machine will work for 00100Δ. Initially, state is q₀ and head points to 0 as:



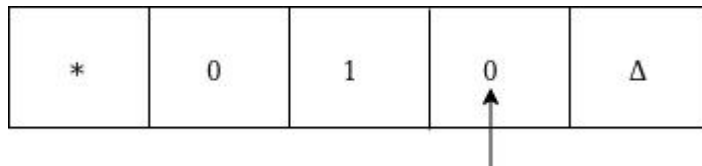
Now replace 0 by * and move right as:



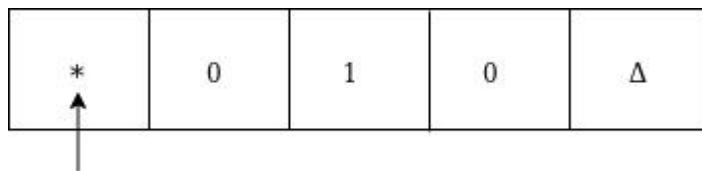
Move right up to Δ as:



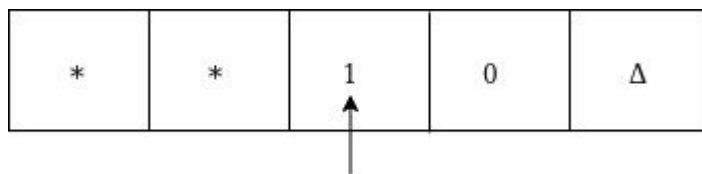
Move left and replace 0 by Δ and move left:



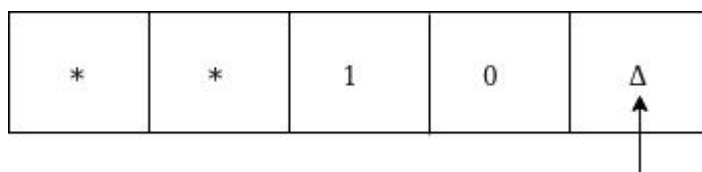
Now move left up to * as:



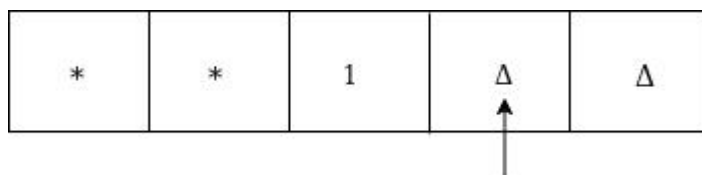
Move right, convert 0 by * and then move right as:



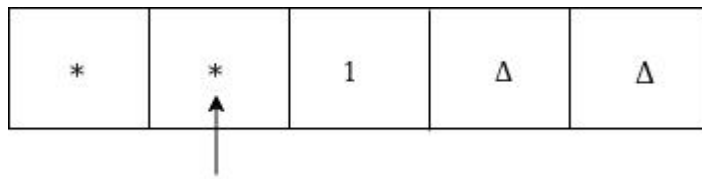
Moved right up to Δ



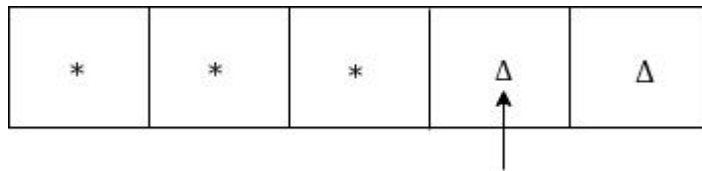
Move left and replace 0 by Δ as:



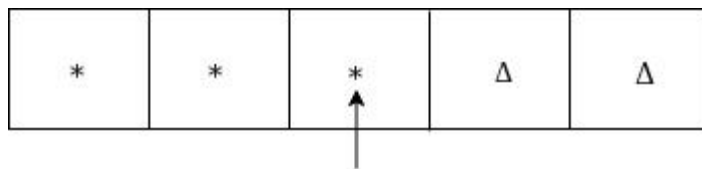
Move left till * as:



Move right and convert 1 to * as:

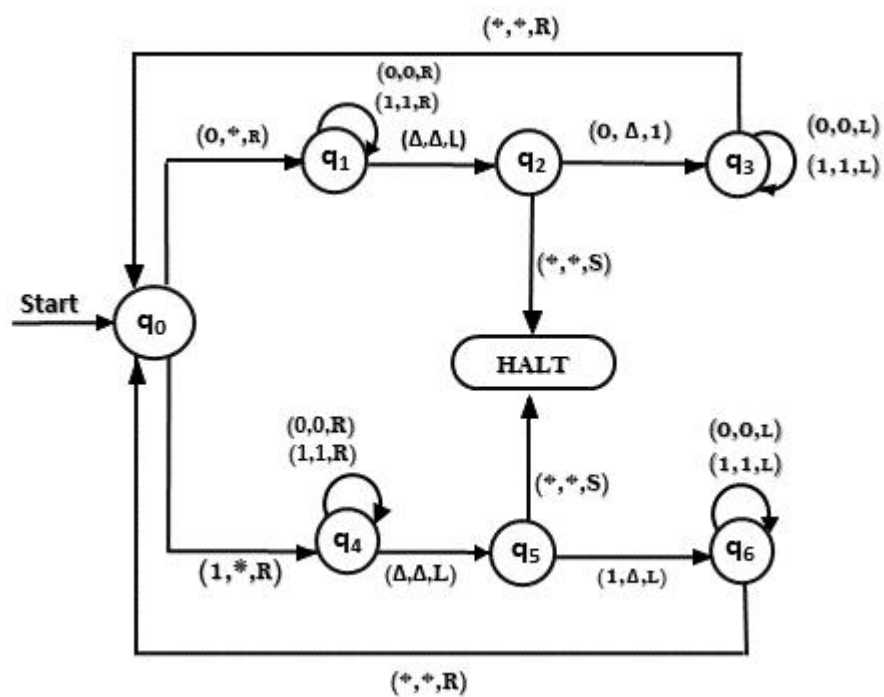


Move left



Since it is *, goto HALT state.

The same TM can be represented by Transition Diagram:



22. Construct TM for the addition function for the unary number system.

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 the addition of two unary numbers.

For example

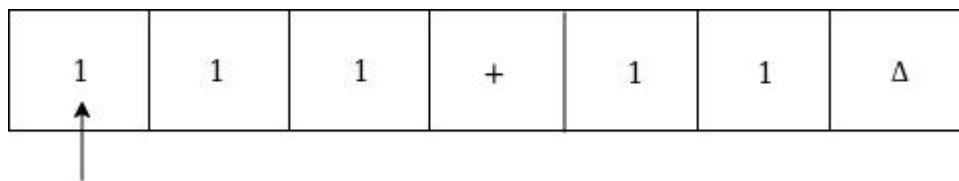
2 + 3
i.e. 11 + 111 = 11111

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

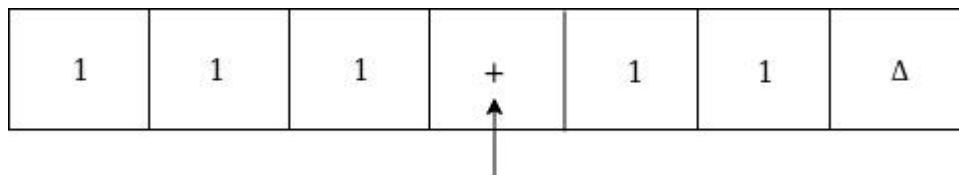
In this case, we simply replace + by 1 and move ahead right for searching end of the string we will convert last 1 to Δ .

Input: 3+2

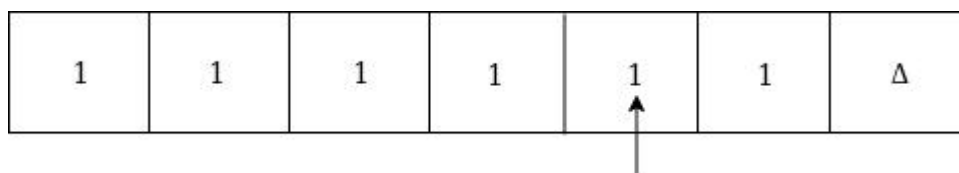
The simulation for 111+11 Δ can be shown as below:



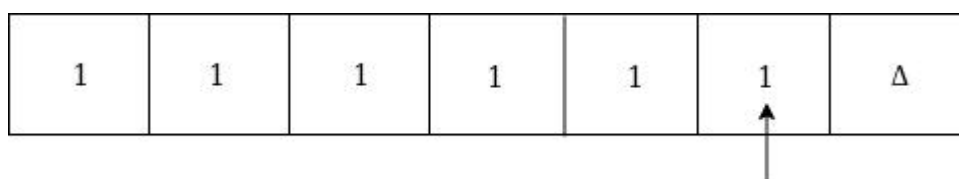
Move right up to + sign as:



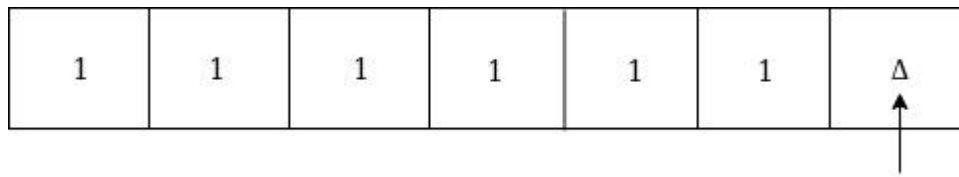
Convert + to 1 and move right as:



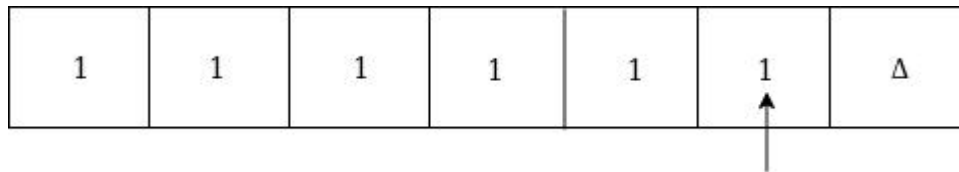
Now, move right



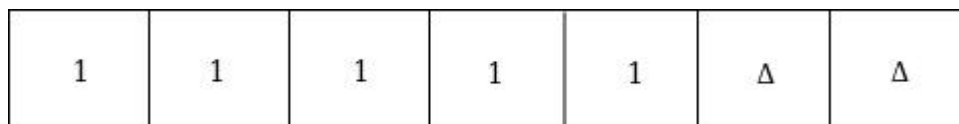
Again move right



Now Δ has encountered, so just move left as:



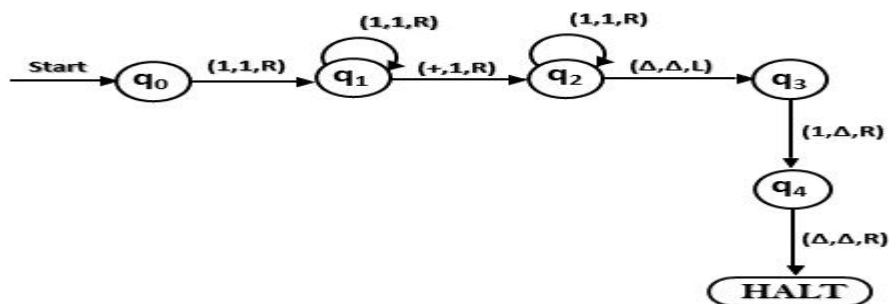
Convert 1 to Δ



Thus the tape now consists of the addition of two unary numbers.

The TM will look like as follows:

Here, we are implementing the function of $f(a + b) = c$. We assume a and b both are non zero elements.



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


Solution:

Here we have certain assumptions as the first number is greater than the second one.
Let us assume that $a = 3$, $b = 2$, so the input tape will be:

1	1	1	-	1	1	Δ
---	---	---	---	---	---	----------


We will move right to - symbol as perform reduction of a number of 1's from the first number. Let us look at the simulation for understanding the logic:

1	1	1	-	1	1	Δ
---	---	---	---	---	---	----------




Move right up to - as:

1	1	1	-	1	1	Δ
---	---	---	---	---	---	----------




Move right and convert 1 to * as:

1	1	1	-	*	1	Δ
---	---	---	---	---	---	----------




Now move left

1	1	1	-	*	1	Δ
---	---	---	---	---	---	----------

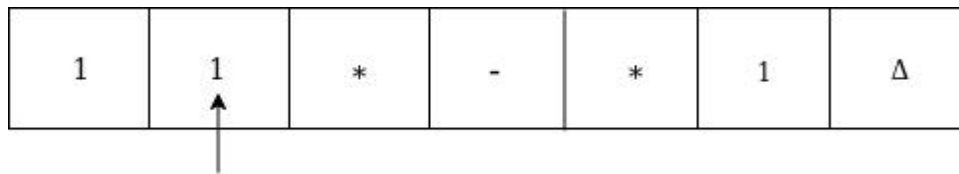


Again move left

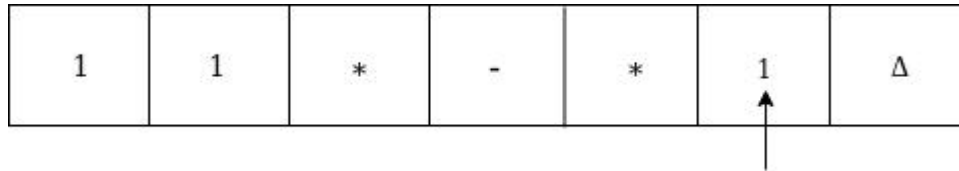
1	1	1	-	*	1	Δ
---	---	---	---	---	---	----------



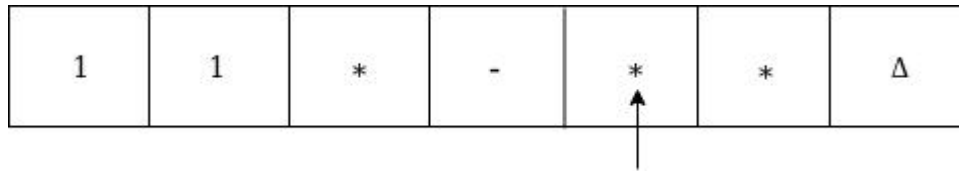
Convert 1 to * and move right-hand



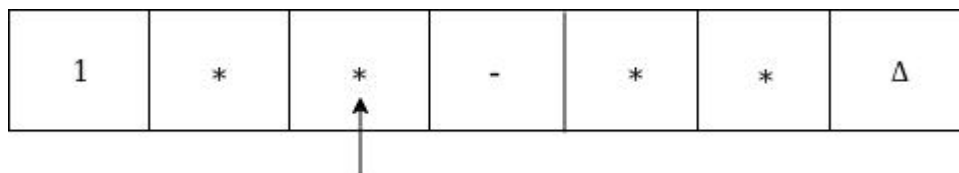
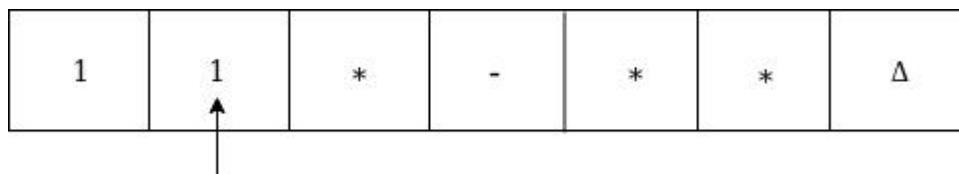
Now move right till 1



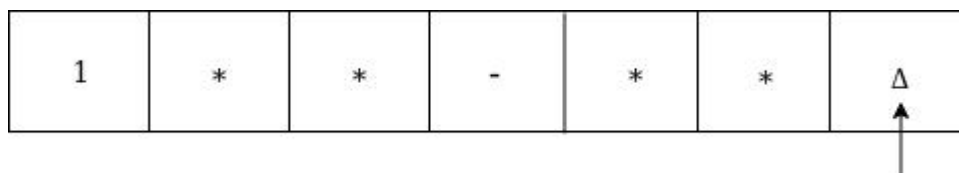
Convert 1 to * and move left



Convert 1 to * and move



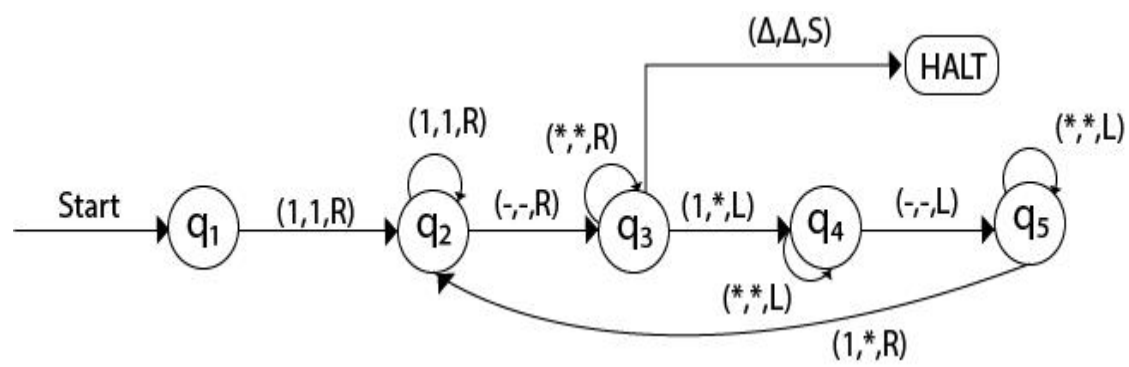
Move right till Δ as:



Now we are in the HALT state.

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

The Turing machine will look like this:



ALL THE BEST