

## Chapter – 3

### PDA Acceptance

A language can be accepted by Pushdown automata using two approaches:

**1. Acceptance by Final State:** The PDA is said to accept its input by the final state if it enters any final state in zero or more moves after reading the entire input.

Let  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$  be a PDA. The language acceptable by the final state can be defined as:

$$L(PDA) = \{w \mid (q_0, w, Z) \vdash^* (p, \varepsilon, \varepsilon), q \in F\}$$

**2. Acceptance by Empty Stack:** On reading the input string from the initial configuration for some PDA, the stack of PDA gets empty.

Let  $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$  be a PDA. The language acceptable by empty stack can be defined as:

1.  $N(PDA) = \{w \mid (q_0, w, Z) \vdash^* (p, \varepsilon, \varepsilon), q \in Q\}$

Equivalence of Acceptance by Final State and Empty Stack

- If  $L = N(P_1)$  for some PDA  $P_1$ , then there is a PDA  $P_2$  such that  $L = L(P_2)$ . That means the language accepted by empty stack PDA will also be accepted by final state PDA.
- If there is a language  $L = L(P_1)$  for some PDA  $P_1$  then there is a PDA  $P_2$  such that  $L = N(P_2)$ . That means language accepted by final state PDA is also acceptable by empty stack PDA.

Example:

Construct a PDA that accepts the language  $L$  over  $\{0, 1\}$  by empty stack which accepts all the string of 0's and 1's in which a number of 0's are twice of number of 1's.

**Solution:**

There are two parts for designing this PDA:

- If 1 comes before any 0's
- If 0 comes before any 1's.

We are going to design the first part i.e. 1 comes before 0's. The logic is that read single 1 and push two 1's onto the stack. Thereafter on reading two 0's, POP two 1's from the stack. The  $\delta$  can be

1.  $\delta(q_0, 1, Z) = (q_0, 11, Z)$  Here Z represents that stack is empty  
 $\delta(q_0, 0, 1) = (q_0, \epsilon)$

Now, consider the second part i.e. if 0 comes before 1's. The logic is that read first 0, push it onto the stack and change state from  $q_0$  to  $q_1$ . [Note that state  $q_1$  indicates that first 0 is read and still second 0 has yet to read].

Being in  $q_1$ , if 1 is encountered then POP 0. Being in  $q_1$ , if 0 is read then simply read that second 0 and move ahead. The  $\delta$  will be:

$\delta(q_0, 0, Z) = (q_1, 0Z)$   
 $\delta(q_1, 0, 0) = (q_1, 0)$   
 $\delta(q_1, 0, Z) = (q_0, \epsilon)$   
(indicate that one 0 and one 1 is already read, so simply read the second 0)  
 $\delta(q_1, 1, 0) = (q_1, \epsilon)$

Now, summarize the complete PDA for given L is:

$\delta(q_0, 1, Z) = (q_0, 11Z)$   
 $\delta(q_0, 0, 1) = (q_1, \epsilon)$   
 $\delta(q_0, 0, Z) = (q_1, 0Z)$   
 $\delta(q_1, 0, 0) = (q_1, 0)$   
 $\delta(q_1, 0, Z) = (q_0, \epsilon)$   
 $\delta(q_0, \epsilon, Z) = (q_0, \epsilon)$  ACCEPT state

**Example:** Define the pushdown automata for language  $\{a^n b^n \mid n > 0\}$  using final state.

**Solution:**  $M =$  where  $Q = \{q_0, q_1, q_2, q_3\}$  and  $\Sigma = \{a, b\}$  and  $\Gamma = \{A, Z\}$  and  $F = \{q_3\}$  and  $\delta$  is given by:

$\delta(q_0, a, Z) = \{(q_1, AZ)\}$

$\delta(q_1, a, A) = \{(q_1, AA)\}$

$\delta(q_1, b, A) = \{(q_2, \epsilon)\}$

$\delta(q_2, b, A) = \{(q_2, \epsilon)\}$

$\delta(q_2, \epsilon, Z) = \{(q_3, Z)\}$

Let us see how this automaton works for aaabbb:

Row	State	Input	$\delta$ (transition function) used	Stack (Leftmost symbol represents top of stack)	State after move
0	q0	aaabbb		Z	
1	q0	<b>a</b> aabbb	$\delta(q0, a, Z) = \{(q1, AZ)\}$	AZ	q1
2	q1	a <b>a</b> abbb	$\delta(q1, a, A) = \{(q1, AA)\}$	AAZ	q1
3	q1	aa <b>a</b> bbb	$\delta(q1, a, A) = \{(q1, AA)\}$	AAAZ	q1
4	q1	aaa <b>b</b> bb	$\delta(q1, b, A) = \{(q2, \epsilon)\}$	AAZ	q2
5	q2	aaab <b>b</b> b	$\delta(q2, b, A) = \{(q2, \epsilon)\}$	AZ	q2
6	q2	aaabb <b>b</b>	$\delta(q2, b, A) = \{(q2, \epsilon)\}$	Z	q2
7	q2	$\epsilon$	$\delta(q2, \epsilon, Z) = \{(q3, \epsilon)\}$	Z	q3

**Explanation:** Initially, the state of automata is q0 and symbol on the stack is Z and the input is aaabbb as shown in row 0. On reading a (shown in bold in row 1), the state will be changed to q1 and it will push symbol A on the stack. On next a (shown in row 2), it will push another symbol A on the stack and remain in state q1. After reading 3 a's, the stack will be AAAZ with A on the top. After reading b (as shown in row 4), it will pop A and move to state q2 and the stack will be AAZ. When all b's are read, the state will be q2 and the stack will be Z. In row 7, on input symbol  $\epsilon$  and Z on the stack, it will move to q3. As the final state q3 has been reached after processing input, the string will be accepted.

Equivalence of Context Free Grammar and PDA:

CFG to PDA Conversion

The first symbol on R.H.S. production must be a terminal symbol. The following steps are used to obtain PDA from CFG is:

**Step 1:** Convert the given productions of CFG into GNF.

**Step 2:** The PDA will only have one state {q}.

**Step 3:** The initial symbol of CFG will be the initial symbol in the PDA.

**Step 4:** For non-terminal symbol, add the following rule:

$$\delta(q, \epsilon, A) = (q, \alpha)$$

Where the production rule is  $A \rightarrow \alpha$

**Step 5:** For each terminal symbols, add the following rule:

$\delta(q, a, a) = (q, \epsilon)$  **for** every terminal symbol

Example 1:

Convert the following grammar to a PDA that accepts the same language.

$S \rightarrow 0S1 \mid A$   
 $A \rightarrow 1A0 \mid S \mid \epsilon$

**Solution:**

The CFG can be first simplified by eliminating unit productions:

$S \rightarrow 0S1 \mid 1S0 \mid \epsilon$

Now we will convert this CFG to GNF:

$S \rightarrow 0SX \mid 1SY \mid \epsilon$   
 $X \rightarrow 1$   
 $Y \rightarrow 0$

The PDA can be:

**R1:**  $\delta(q, \epsilon, S) = \{(q, 0SX) \mid (q, 1SY) \mid (q, \epsilon)\}$

**R2:**  $\delta(q, \epsilon, X) = \{(q, 1)\}$

**R3:**  $\delta(q, \epsilon, Y) = \{(q, 0)\}$

**R4:**  $\delta(q, 0, 0) = \{(q, \epsilon)\}$

**R5:**  $\delta(q, 1, 1) = \{(q, \epsilon)\}$

Example 2:

Construct PDA for the given CFG, and test whether 0104 is acceptable by this PDA.

$S \rightarrow 0BB$   
 $B \rightarrow 0S \mid 1S \mid 0$

**Solution:**

The PDA can be given as:

$A = \{(q), (0, 1), (S, B, 0, 1), \delta, q, S, ?\}$

The production rule  $\delta$  can be:

**R1:**  $\delta(q, \varepsilon, S) = \{(q, 0BB)\}$   
**R2:**  $\delta(q, \varepsilon, B) = \{(q, 0S) \mid (q, 1S) \mid (q, 0)\}$   
**R3:**  $\delta(q, 0, 0) = \{(q, \varepsilon)\}$   
**R4:**  $\delta(q, 1, 1) = \{(q, \varepsilon)\}$

Testing 010<sup>4</sup> i.e. 010000 against PDA:

$\delta(q, 010000, S) \vdash \delta(q, 010000, 0BB)$   
 $\vdash \delta(q, 10000, BB) \quad R1$   
 $\vdash \delta(q, 10000, 1SB) \quad R3$   
 $\vdash \delta(q, 0000, SB) \quad R2$   
 $\vdash \delta(q, 0000, 0BBB) \quad R1$   
 $\vdash \delta(q, 000, BBB) \quad R3$   
 $\vdash \delta(q, 000, 0BB) \quad R2$   
 $\vdash \delta(q, 00, BB) \quad R3$   
 $\vdash \delta(q, 00, 0B) \quad R2$   
 $\vdash \delta(q, 0, B) \quad R3$   
 $\vdash \delta(q, 0, 0) \quad R2$   
 $\vdash \delta(q, \varepsilon) \quad R3$   
 ACCEPT

Thus 010<sup>4</sup> is accepted by the PDA.

Example 3:

Draw a PDA for the CFG given below:

$S \rightarrow aSb$   
 $S \rightarrow a \mid b \mid \varepsilon$

**Solution:**

The PDA can be given as:

$P = \{(q), (a, b), (S, a, b, z0), \delta, q, z0, q\}$

The mapping function  $\delta$  will be:

**R1:**  $\delta(q, \varepsilon, S) = \{(q, aSb)\}$   
**R2:**  $\delta(q, \varepsilon, S) = \{(q, a) \mid (q, b) \mid (q, \varepsilon)\}$   
**R3:**  $\delta(q, a, a) = \{(q, \varepsilon)\}$   
**R4:**  $\delta(q, b, b) = \{(q, \varepsilon)\}$   
**R5:**  $\delta(q, \varepsilon, z0) = \{(q, \varepsilon)\}$

**Simulation:** Consider the string aaabbb

$\delta(q, \epsilon aaabb, S) \vdash \delta(q, aaabb, aSb)$	R3
$\vdash \delta(q, \epsilon aabb, Sb)$	R1
$\vdash \delta(q, aabb, aSbb)$	R3
$\vdash \delta(q, \epsilon abb, Sbb)$	R2
$\vdash \delta(q, abb, abb)$	R3
$\vdash \delta(q, bb, bb)$	R4
$\vdash \delta(q, b, b)$	R4
$\vdash \delta(q, \epsilon, z0)$	R5
$\vdash \delta(q, \epsilon)$	
ACCEPT	

## Turing Machine

Turing machine was invented in 1936 by **Alan Turing**. It is an accepting device which accepts Recursive Enumerable Language generated by type 0 grammar.

There are various features of the Turing machine:

1. It has an external memory which remembers arbitrary long sequence of input.
2. It has unlimited memory capability.
3. The model has a facility by which the input at left or right on the tape can be read easily.
4. The machine can produce a certain output based on its input. Sometimes it may be required that the same input has to be used to generate the output. So in this machine, the distinction between input and output has been removed. Thus a common set of alphabets can be used for the Turing machine.

## Introduction of Turing machine

A Turing machine can be defined as a collection of 7 components:

**Q**: the finite set of states

$\Sigma$ : the finite set of input symbols

**T**: the tape symbol

**q0**: the initial state

**F**: a set of final states

**B**: a blank symbol used as a end marker for input

$\delta$ : a transition or mapping function.

The mapping function shows the mapping from states of finite automata and input symbol on the tape to the next states, external symbols and the direction for moving the tape head. This is known as a triple or a program for turing machine.

$(q0, a) \rightarrow (q1, A, R)$

That means in  $q_0$  state, if we read symbol 'a' then it will go to state  $q_1$ , replaced a by X and move ahead right(R stands for right).

Example:

Construct TM for the language  $L = \{0^n 1^n\}$  where  $n \geq 1$ .

### Solution:

We have already solved this problem by PDA. In PDA, we have a stack to remember the previous symbol. The main advantage of the Turing machine is we have a tape head which can be moved forward or backward, and the input tape can be scanned.

The simple logic which we will apply is read out each '0' mark it by A and then move ahead along with the input tape and find out 1 convert it to B. Now, repeat this process for all a's and b's.

Now we will see how this turing machine work for 0011.

The simulation for 0011 can be shown as below:

0	0	1	1	$\Delta$	-----
---	---	---	---	----------	-------

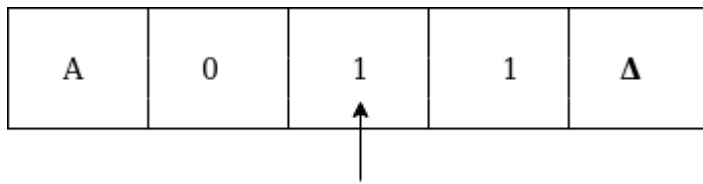
Now, we will see how this turing machine will works for 0011. Initially, state is  $q_0$  and head points to 0 as:

0	0	1	1	$\Delta$
↑				

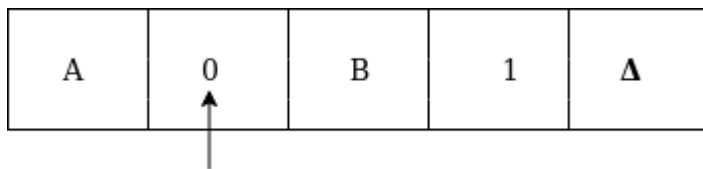
The move will be  $\delta(q_0, 0) = \delta(q_1, A, R)$  which means it will go to state  $q_1$ , replaced 0 by A and head will move to the right as:

A	0	1	1	$\Delta$
	↑			

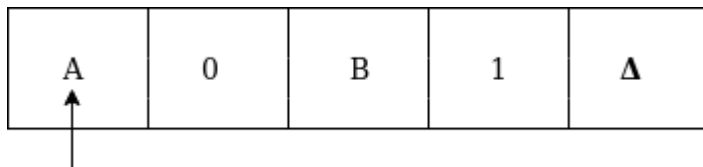
The move will be  $\delta(q_1, 0) = \delta(q_1, 0, R)$  which means it will not change any symbol, remain in the same state and move to the right as:



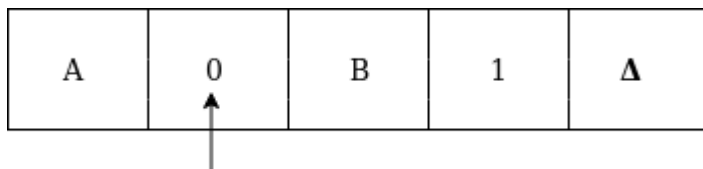
The move will be  $\delta(q_1, 1) = \delta(q_2, B, L)$  which means it will go to state  $q_2$ , replaced 1 by B and head will move to left as:



Now move will be  $\delta(q_2, 0) = \delta(q_2, 0, L)$  which means it will not change any symbol, remain in the same state and move to left as:

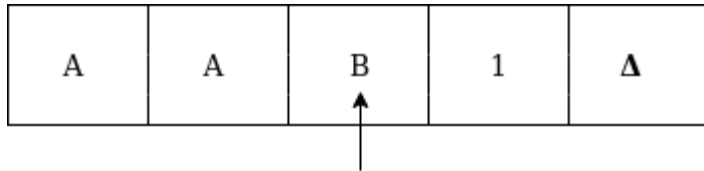


The move will be  $\delta(q_2, A) = \delta(q_0, A, R)$ , it means will go to state  $q_0$ , replaced A by A and head will move to the right as:

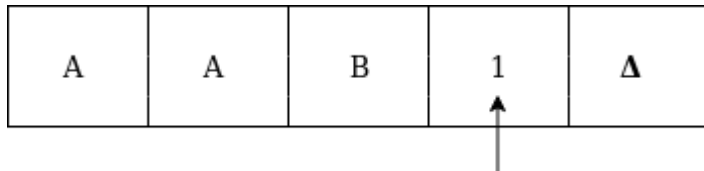


The move will be  $\delta(q_0, 0) = \delta(q_1, A, R)$  which means it will go to state  $q_1$ , replaced 0 by A, and head will move to right as:

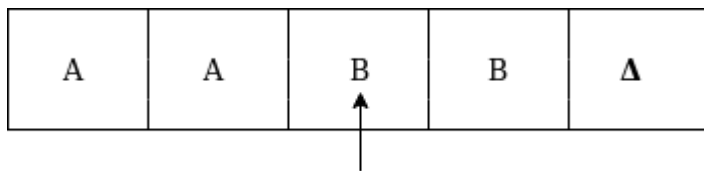




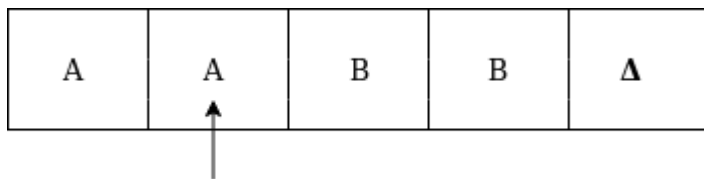
The move will be  $\delta(q_1, B) = \delta(q_1, B, R)$  which means it will not change any symbol, remain in the same state and move to right as:



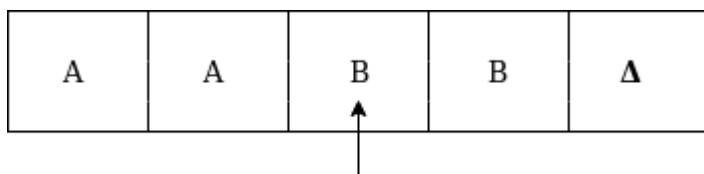
The move will be  $\delta(q_1, 1) = \delta(q_2, B, L)$  which means it will go to state  $q_2$ , replaced 1 by B and head will move to left as:



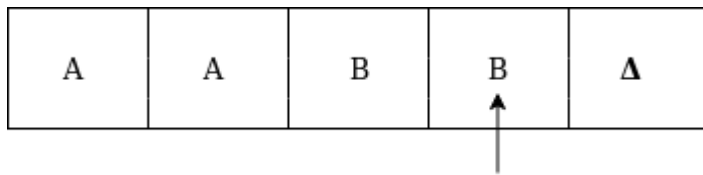
The move  $\delta(q_2, B) = (q_2, B, L)$  which means it will not change any symbol, remain in the same state and move to left as:



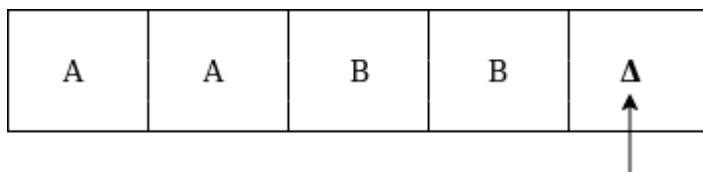
Now immediately before B is A that means all the 0's are marked by A. So we will move right to ensure that no 1 is present. The move will be  $\delta(q_2, A) = (q_0, A, R)$  which means it will go to state  $q_0$ , will not change any symbol, and move to right as:



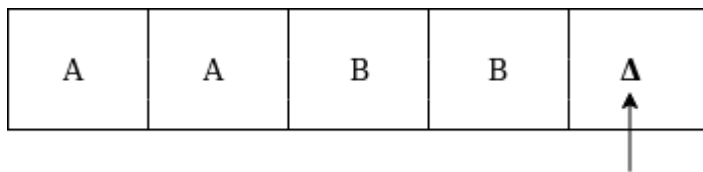
The move  $\delta(q_0, B) = (q_3, B, R)$  which means it will go to state  $q_3$ , will not change any symbol, and move to right as:



The move  $\delta(q_3, B) = (q_3, B, R)$  which means it will not change any symbol, remain in the same state and move to right as:



The move  $\delta(q_3, \Delta) = (q_4, \Delta, R)$  which means it will go to state  $q_4$  which is the HALT state and HALT state is always an accept state for any TM.



The same TM can be represented by Transition Diagram:

## Design of TM

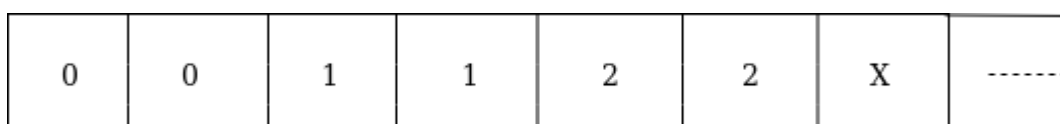
Example 1:

Construct a TM for the language  $L = \{0^n 1^n 2^n\}$  where  $n \geq 1$

### Solution:

$L = \{0^n 1^n 2^n \mid n \geq 1\}$  represents language where we use only 3 character, i.e., 0, 1 and 2. In this, some number of 0's followed by an equal number of 1's and then followed by an equal number of 2's. Any type of string which falls in this category will be accepted by this language.

The simulation for 001122 can be shown as below:



Now, we will see how this Turing machine will work for 001122. Initially, state is  $q_0$  and head points to 0 as:

0	0	1	1	2	2	X
---	---	---	---	---	---	---

↑

The move will be  $\delta(q_0, 0) = \delta(q_1, A, R)$  which means it will go to state  $q_1$ , replaced 0 by A and head will move to the right as:

A	0	1	1	2	2	X
---	---	---	---	---	---	---

↑

The move will be  $\delta(q_1, 0) = \delta(q_1, 0, R)$  which means it will not change any symbol, remain in the same state and move to the right as:

A	0	1	1	2	2	X
---	---	---	---	---	---	---

↑

The move will be  $\delta(q_1, 1) = \delta(q_2, B, R)$  which means it will go to state  $q_2$ , replaced 1 by B and head will move to right as:

A	0	B	1	2	2	X
---	---	---	---	---	---	---

↑


The move will be  $\delta(q_2, 1) = \delta(q_2, 1, R)$  which means it will not change any symbol, remain in the same state and move to right as:

A	0	B	1	2	2	X
---	---	---	---	---	---	---

↑


The move will be  $\delta(q_2, 2) = \delta(q_3, C, R)$  which means it will go to state  $q_3$ , replaced 2 by C and head will move to right as:

A	0	B	1	C	2	X
---	---	---	---	---	---	---




Now move  $\delta(q_3, 2) = \delta(q_3, 2, L)$  and  $\delta(q_3, C) = \delta(q_3, C, L)$  and  $\delta(q_3, 1) = \delta(q_3, 1, L)$  and  $\delta(q_3, B) = \delta(q_3, B, L)$  and  $\delta(q_3, 0) = \delta(q_3, 0, L)$ , and then move  $\delta(q_3, A) = \delta(q_0, A, R)$ , it means will go to state  $q_0$ , replaced A by A and head will move to right as:

A	0	B	1	C	2	X
---	---	---	---	---	---	---




The move will be  $\delta(q_0, 0) = \delta(q_1, A, R)$  which means it will go to state  $q_1$ , replaced 0 by A, and head will move to right as:

A	A	B	1	C	2	X
---	---	---	---	---	---	---




The move will be  $\delta(q_1, B) = \delta(q_1, B, R)$  which means it will not change any symbol, remain in the same state and move to right as:

A	A	B	1	C	2	X
---	---	---	---	---	---	---

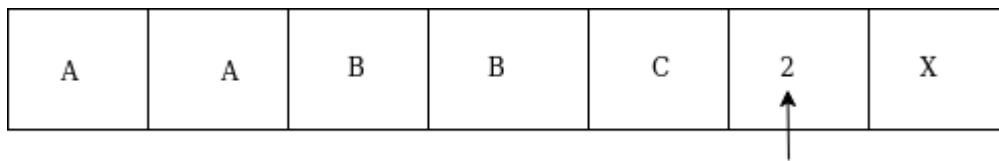


The move will be  $\delta(q_1, 1) = \delta(q_2, B, R)$  which means it will go to state  $q_2$ , replaced 1 by B and head will move to right as:

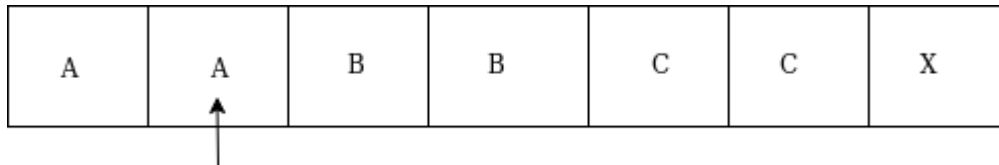
A	A	B	B	C	2	X
---	---	---	---	---	---	---



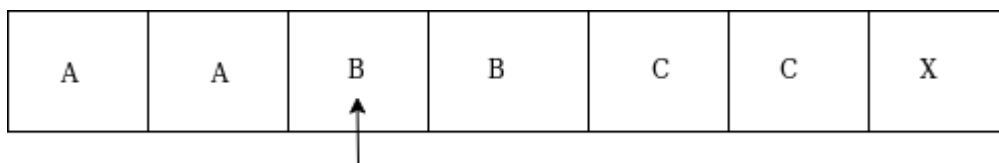
The move will be  $\delta(q_2, C) = \delta(q_2, C, R)$  which means it will not change any symbol, remain in the same state and move to right as:



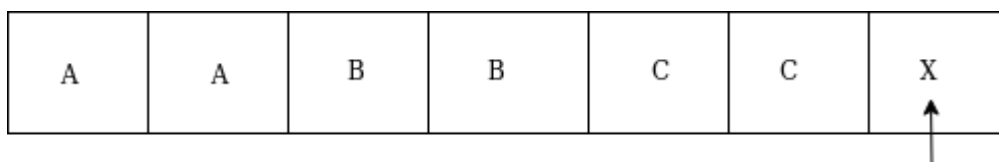
The move will be  $\delta(q_2, 2) = \delta(q_3, C, L)$  which means it will go to state  $q_3$ , replaced 2 by C and head will move to left until we reached A as:



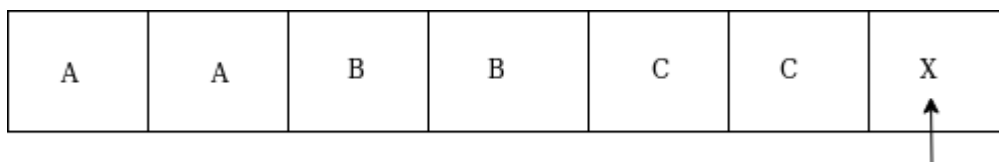
immediately before B is A that means all the 0's are marked by A. So we will move right to ensure that no 1 or 2 is present. The move will be  $\delta(q_3, B) = (q_4, B, R)$  which means it will go to state  $q_4$ , will not change any symbol, and move to right as:



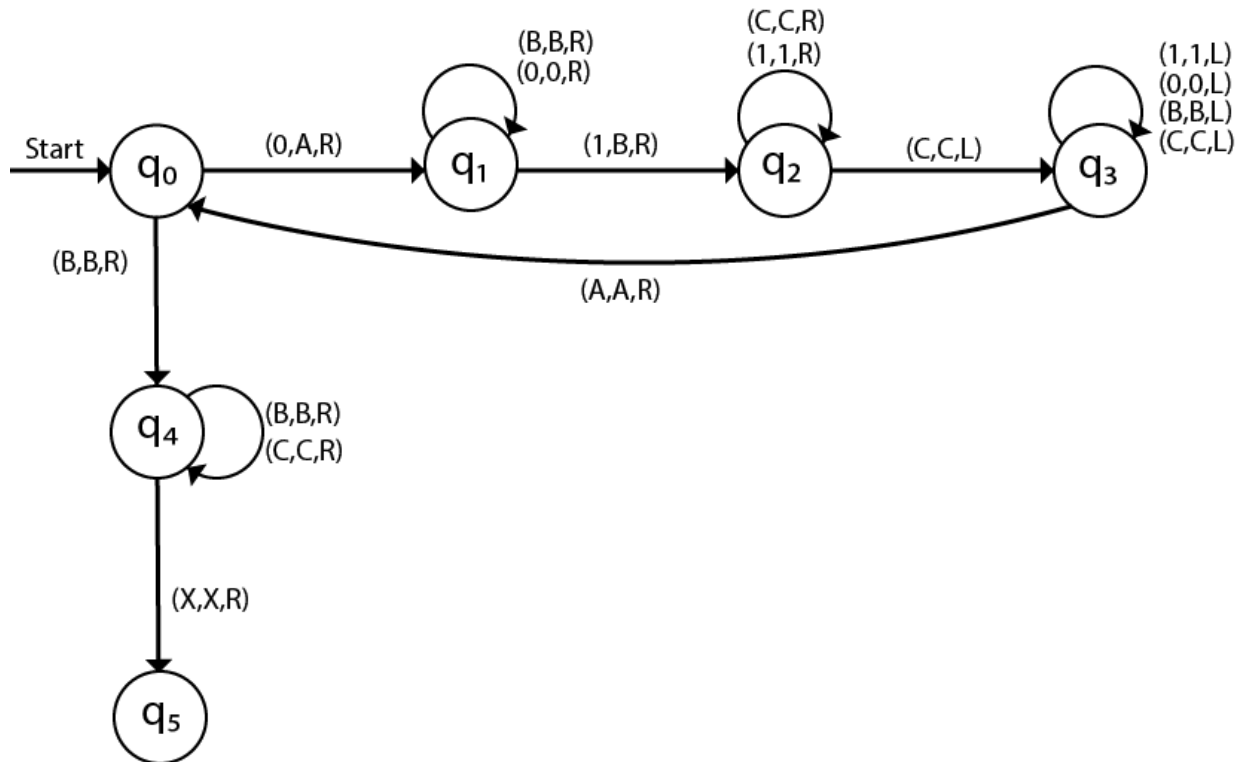
The move will be  $(q_4, B) = \delta(q_4, B, R)$  and  $(q_4, C) = \delta(q_4, C, R)$  which means it will not change any symbol, remain in the same state and move to right as:



The move  $\delta(q_4, X) = (q_5, X, R)$  which means it will go to state  $q_5$  which is the HALT state and HALT state is always an accept state for any TM.



The same TM can be represented by Transition Diagram:



Example 2:

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

**Solution:**

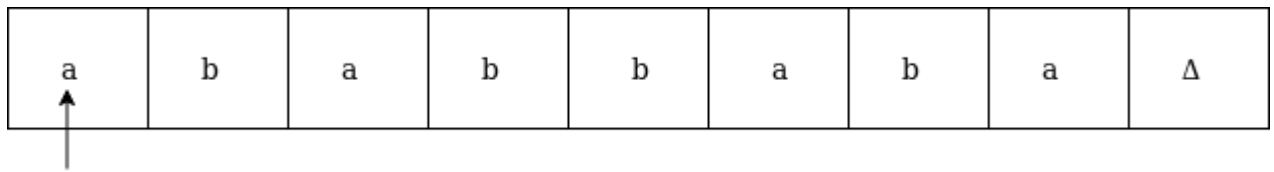
Firstly we read the first symbol from the left and then we compare it with the first symbol from right to check whether it is the same.

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.

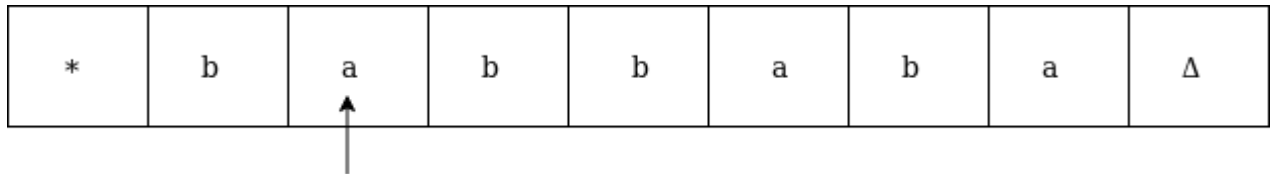
Suppose the string is ababbabaΔ. The simulation for ababbabaΔ can be shown as follows:

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

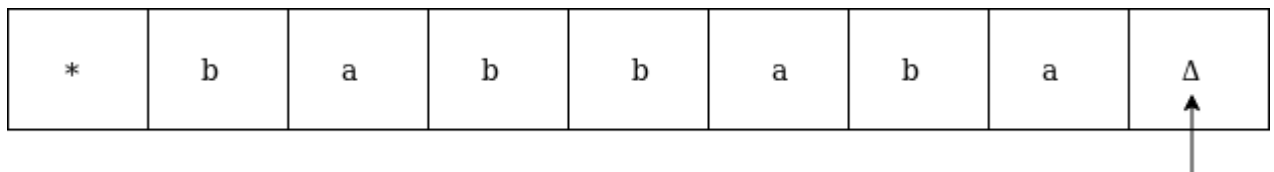
Now, we will see how this Turing machine will work for ababbabaΔ. Initially, state is q<sub>0</sub> and head points to a as:



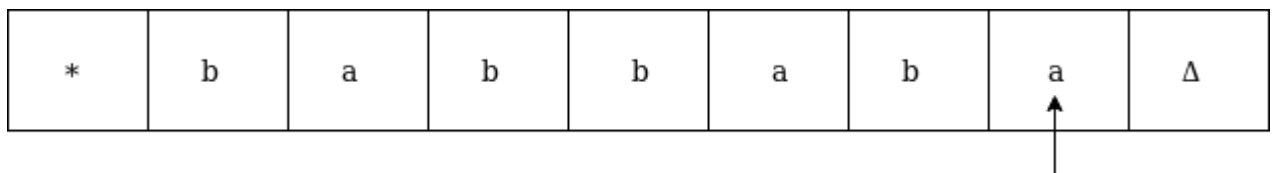
We will mark it by \* and move to right end in search of a as:



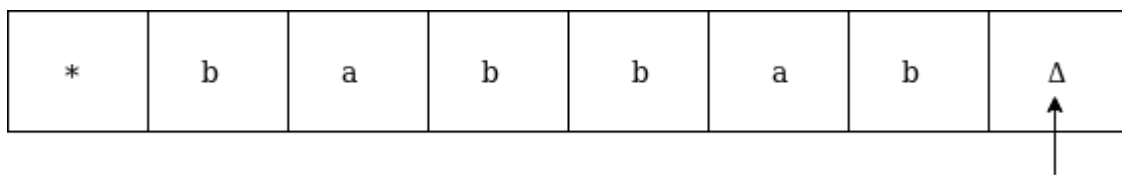
We will move right up to  $\Delta$  as:



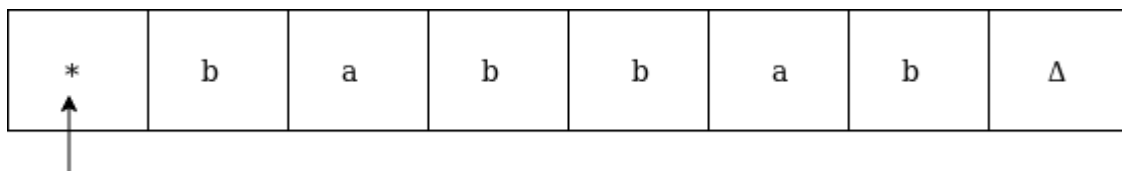
We will move left and check if it is a:



It is 'a' so replace it by  $\Delta$  and move left as:




Now move to left up to \* as:




Move right and read it

*	b	a	b	b	a	b	$\Delta$
---	---	---	---	---	---	---	----------




Now convert b by \* and move right as:

*	*	a	b	b	a	b	$\Delta$
---	---	---	---	---	---	---	----------




Move right up to  $\Delta$  in search of b as:

*	*	a	b	b	a	b	$\Delta$
---	---	---	---	---	---	---	----------




Move left, if the symbol is b then convert it into  $\Delta$  as:

*	*	a	b	b	a	$\Delta$
---	---	---	---	---	---	----------




Now move left until \* as:

*	*	a	b	b	a	$\Delta$
---	---	---	---	---	---	----------



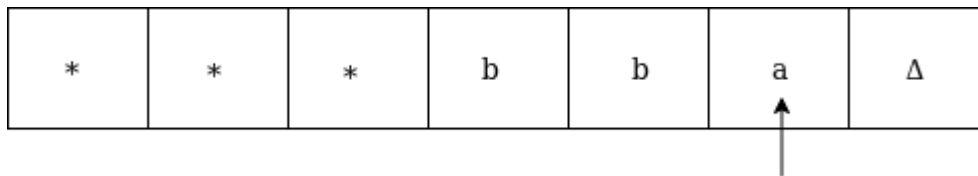
Replace a by \* and move right up to  $\Delta$  as:

*	*	*	b	b	a	$\Delta$
---	---	---	---	---	---	----------

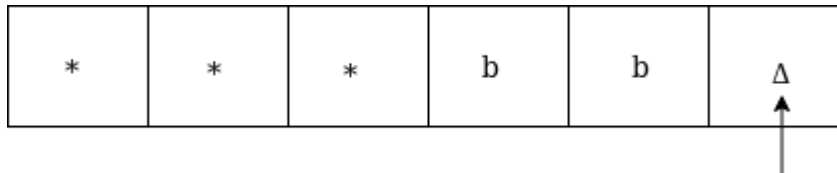


We will move left and check if it is a, then replace it by  $\Delta$  as:

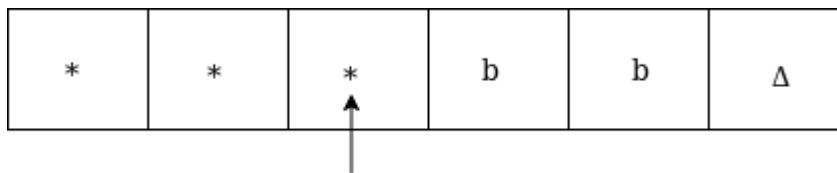




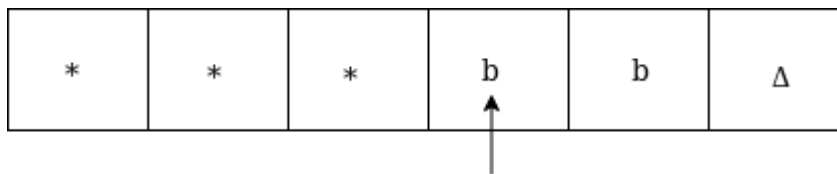
It is 'a' so replace it by  $\Delta$  as:



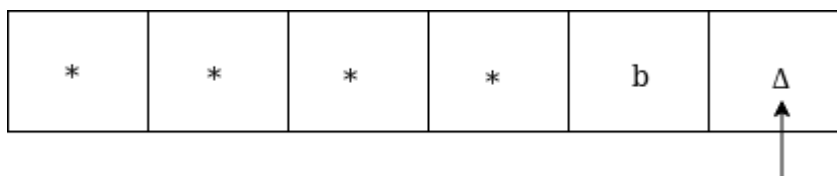
Now move left until \*



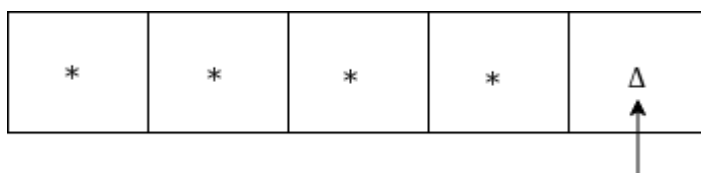
Now move right as:



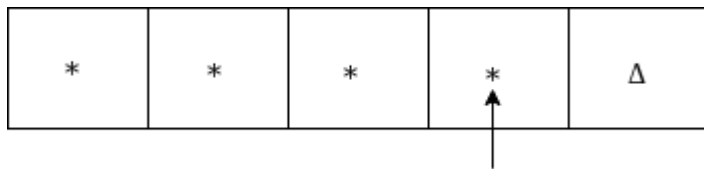
Replace b by \* and move right up to  $\Delta$  as:



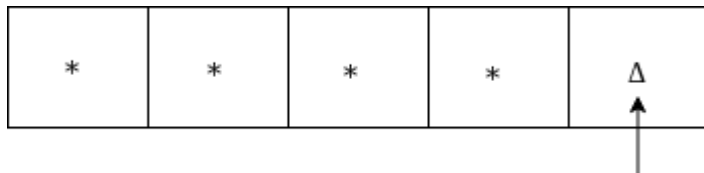
Move left, if the left symbol is b, replace it by  $\Delta$  as:



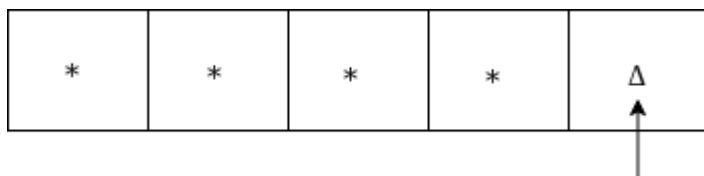
Move left till \*



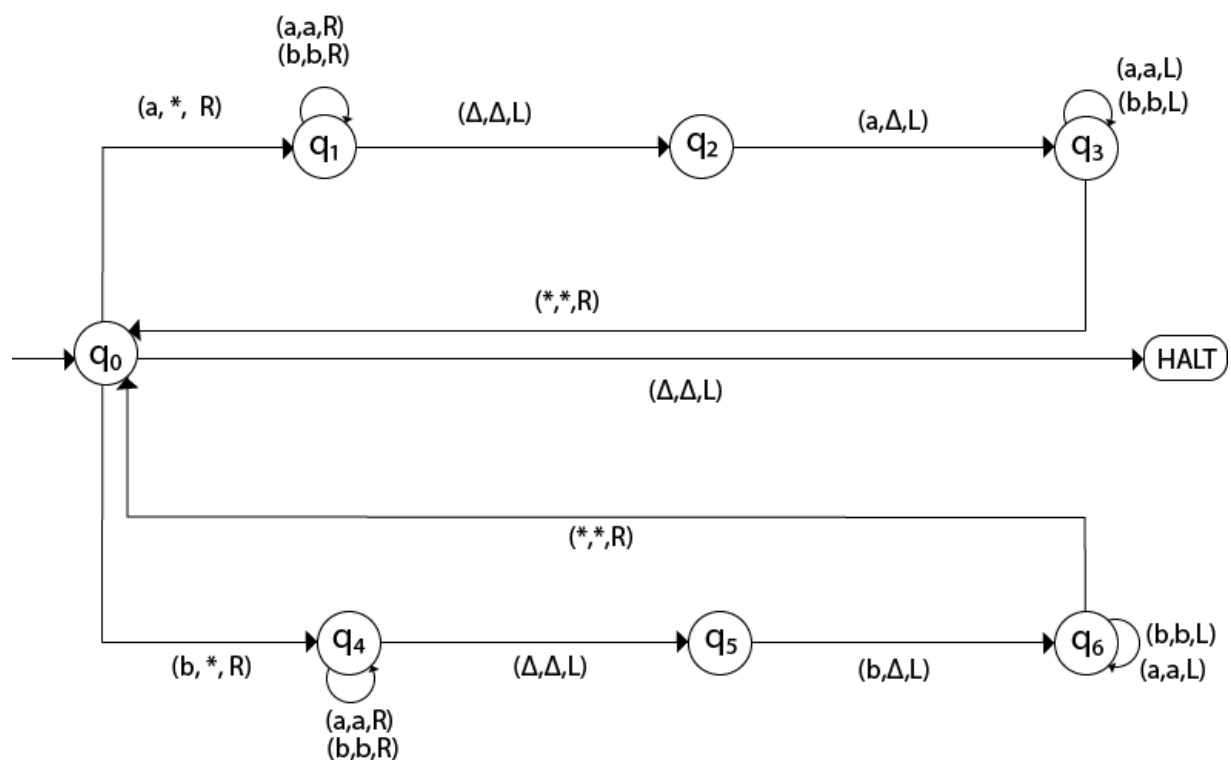
Move right and check whether it is  $\Delta$



Go to HALT state



The same TM can be represented by Transition Diagram:



Example 3:

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

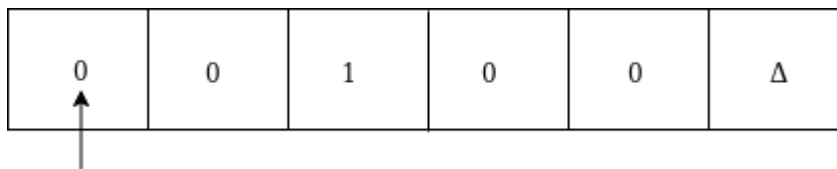
### Solution:

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.

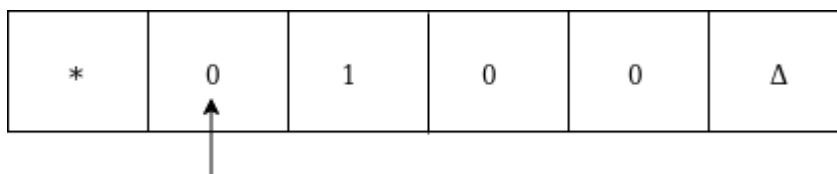
Again we compare the second symbol from left with the second symbol from right. We repeat this process for all the symbols. If we found any symbol not matching, we lead the machine to HALT state.

Suppose the string is 00100 $\Delta$ . The simulation for 00100 $\Delta$  can be shown as follows:

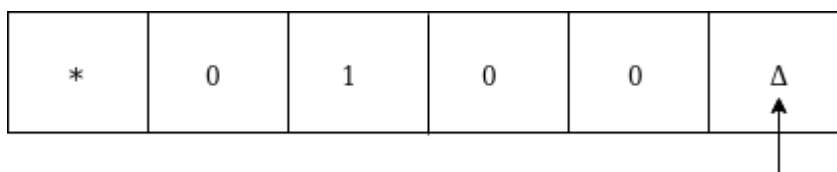
Now, we will see how this Turing machine will work for 00100 $\Delta$ . Initially, state is  $q_0$  and head points to 0 as:



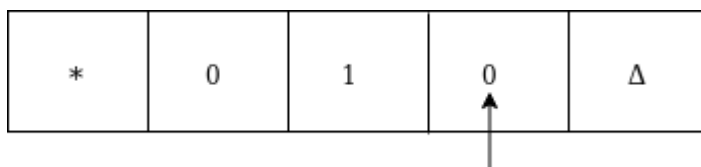
Now replace 0 by \* and move right as:



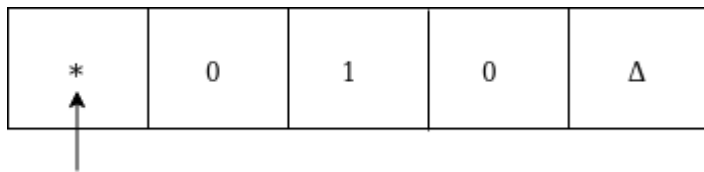
Move right up to  $\Delta$  as:



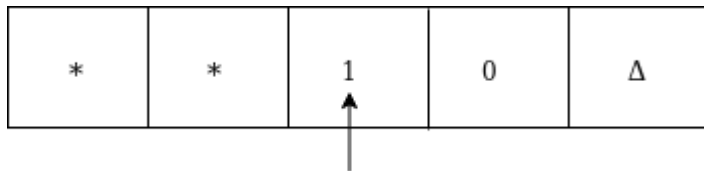
Move left and replace 0 by  $\Delta$  and move left:



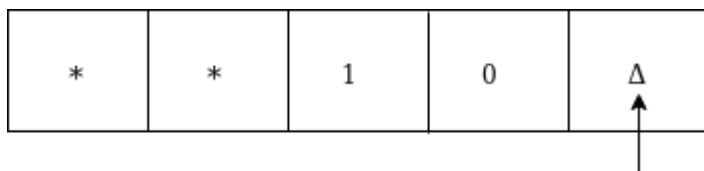
Now move left up to \* as:



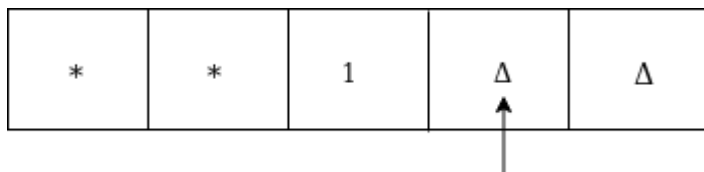
Move right, convert 0 by \* and then move right as:



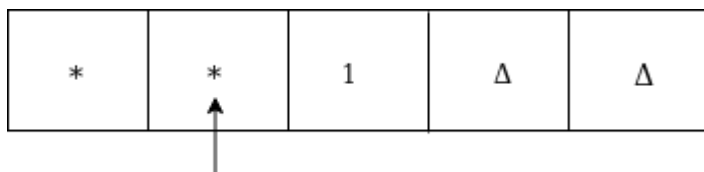
Moved right up to  $\Delta$



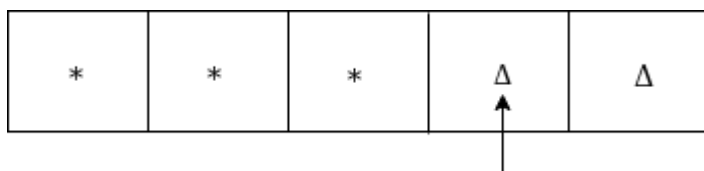
Move left and replace 0 by  $\Delta$  as:



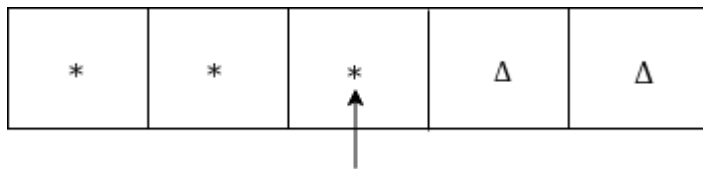
Move left till \* as:



Move right and convert 1 to \* as:

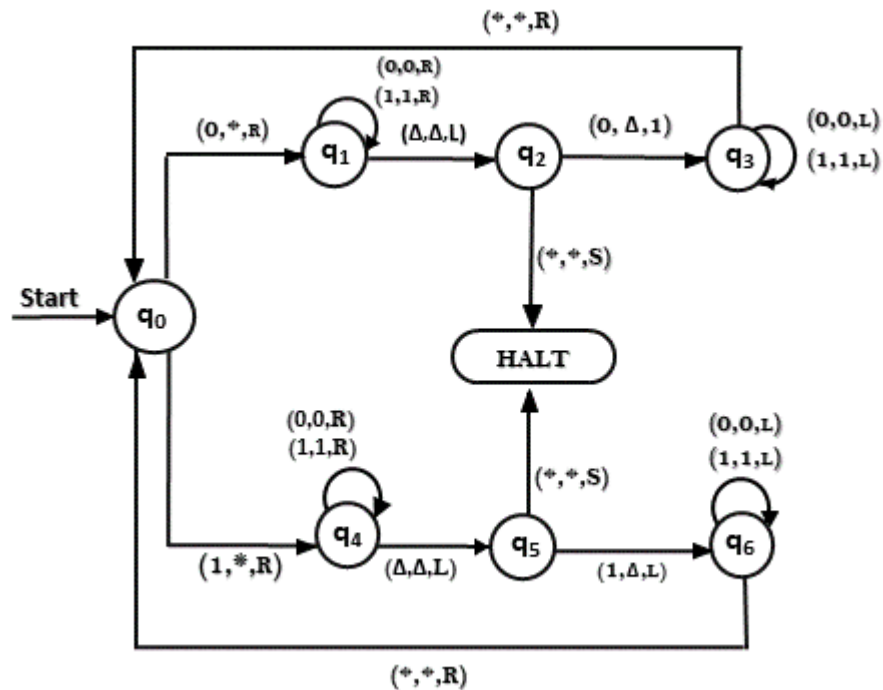


Move left



Since it is \*, goto HALT state.

The same TM can be represented by Transition Diagram:



Example 4:

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

$$2 + 3$$

$$\text{i.e. } 11 + 111 = 11111$$

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

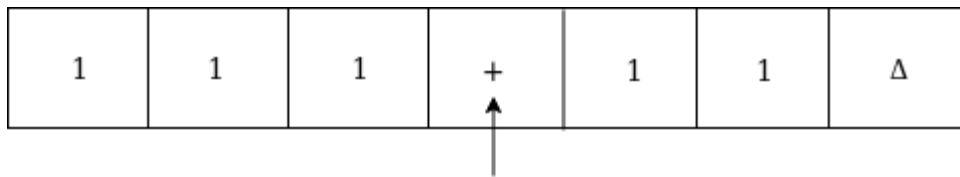
In this case, we simply replace + by 1 and move ahead right for searching end of the string we will convert last 1 to  $\Delta$ .

**Input:** 3+2

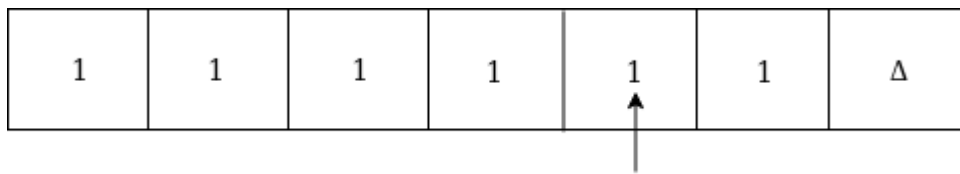
The simulation for 111+11 $\Delta$  can be shown as below:



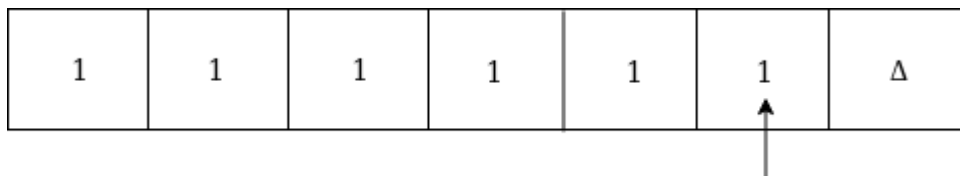
Move right up to + sign as:



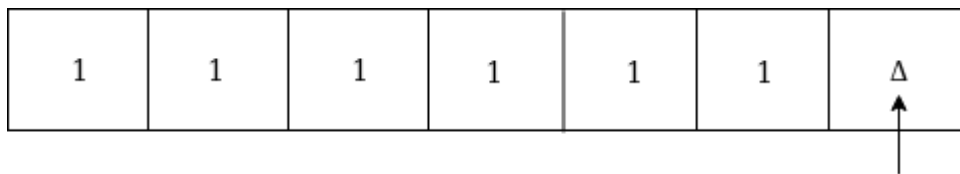
Convert + to 1 and move right as:



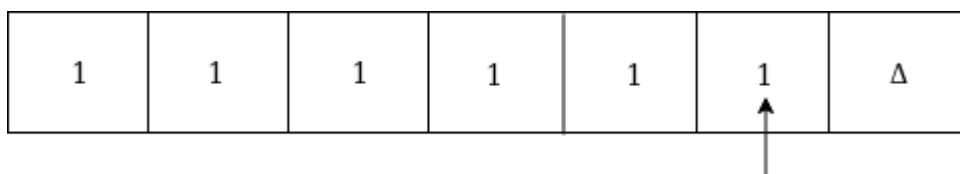
Now, move right



Again move right



Now  $\Delta$  has encountered, so just move left as:



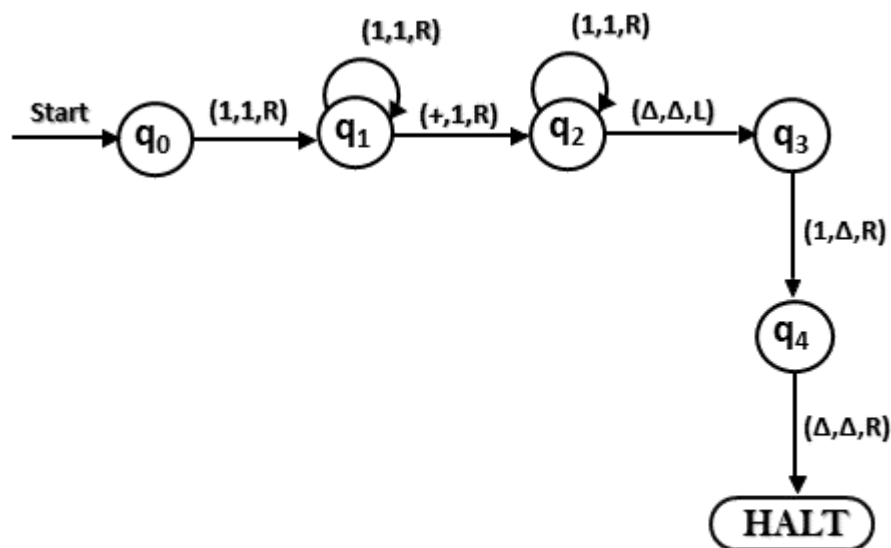
Convert 1 to  $\Delta$

1	1	1	1	1	$\Delta$	$\Delta$
---	---	---	---	---	----------	----------

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.



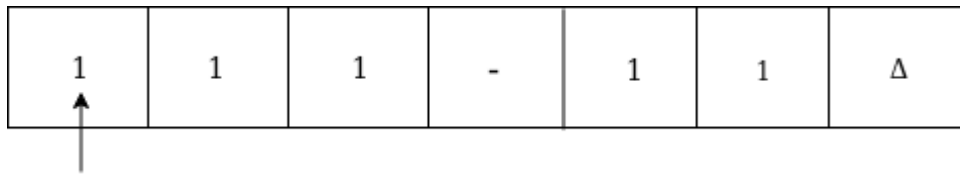
Example 5:

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

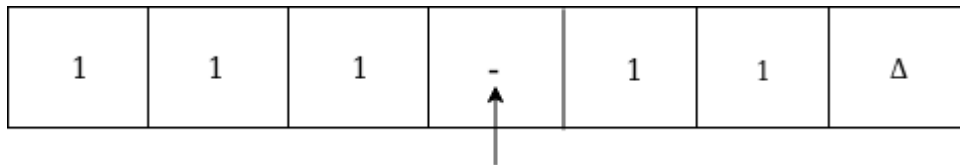
**Solution:** Here we have certain assumptions as the first number is greater than the second one. Let us assume that  $a = 3$ ,  $b = 2$ , so the input tape will be:

1	1	1	-	1	1	$\Delta$
---	---	---	---	---	---	----------

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:



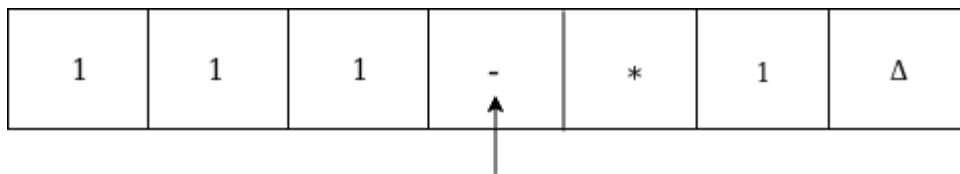
Move right up to - as:



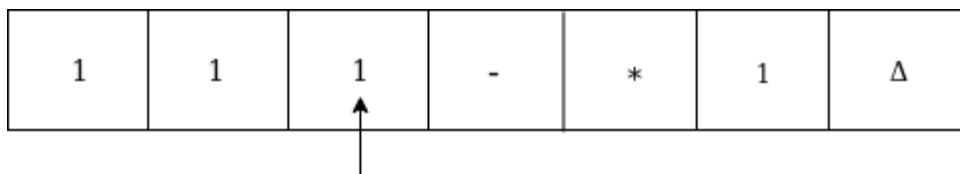
Move right and convert 1 to \* as:



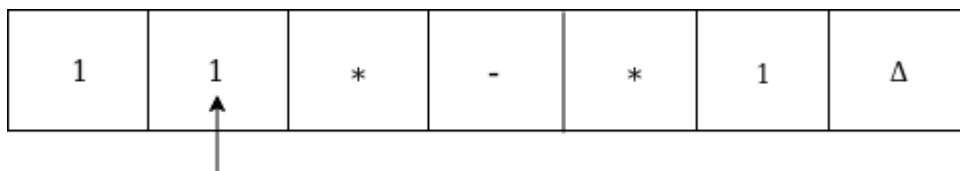
Now move left



Again move left

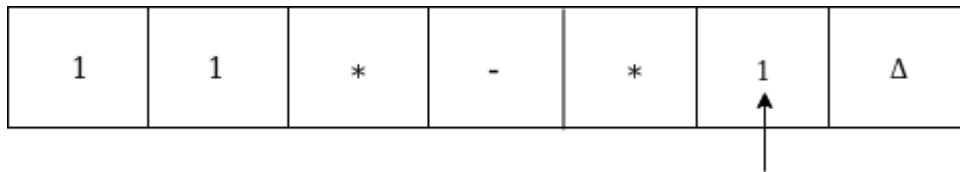


Convert 1 to \* and move right-hand

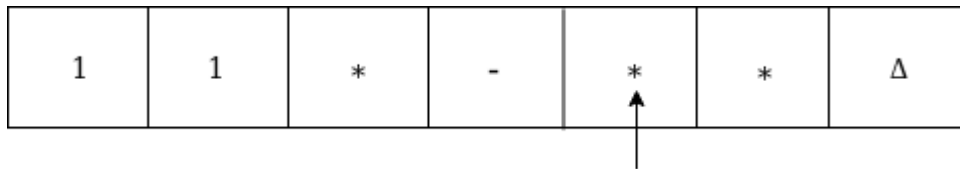


Now move right till 1

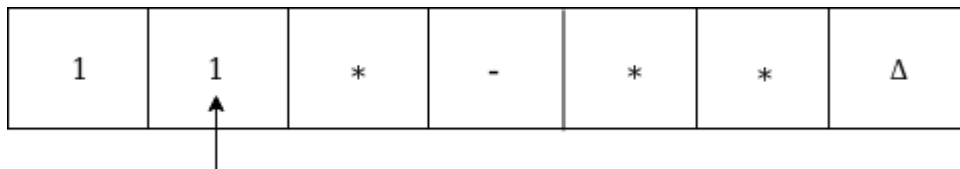




Convert 1 to \* and move left



Convert 1 to \* and move



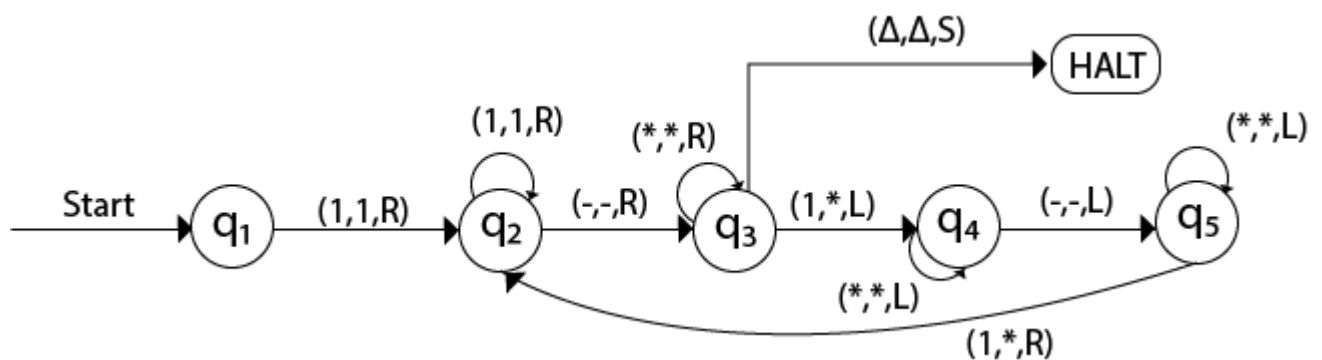
Move right till  $\Delta$  as:



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:



## **Types of Turing Machine:**

### **1. Multiple track Turing Machine:**

- A k-track Turing machine(for some  $k > 0$ ) has k-tracks and one R/W head that reads and writes all of them one by one.
- A k-track Turing Machine can be simulated by a single track Turing machine

### **2. Two-way infinite Tape Turing Machine:**

- Infinite tape of two-way infinite tape Turing machine is unbounded in both directions left and right.
- Two-way infinite tape Turing machine can be simulated by one-way infinite Turing machine(standard Turing machine).

### **3. Multi-tape Turing Machine:**

- It has multiple tapes and is controlled by a single head.
- The Multi-tape Turing machine is different from k-track Turing machine but expressive power is the same.
- Multi-tape Turing machine can be simulated by single-tape Turing machine.

### **4. Multi-tape Multi-head Turing Machine:**

- The multi-tape multi-head Turing machine has multiple tapes and multiple heads
- Each tape is controlled by a separate head
- Multi-Tape Multi-head Turing machine can be simulated by a standard Turing machine.

### **5. Multi-dimensional Tape Turing Machine:**

- It has multi-dimensional tape where the head can move in any direction that is left, right, up or down.
- Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine

### **6. Multi-head Turing Machine:**

- A multi-head Turing machine contains two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently.
- Multi-head Turing machine can be simulated by a single head Turing machine.

### **7. Non-deterministic Turing Machine:**

- A non-deterministic Turing machine has a single, one-way infinite tape.

- For a given state and input symbol has at least one choice to move (finite number of choices for the next move), each choice has several choices of the path that it might follow for a given input string.
- A non-deterministic Turing machine is equivalent to the deterministic Turing machine.