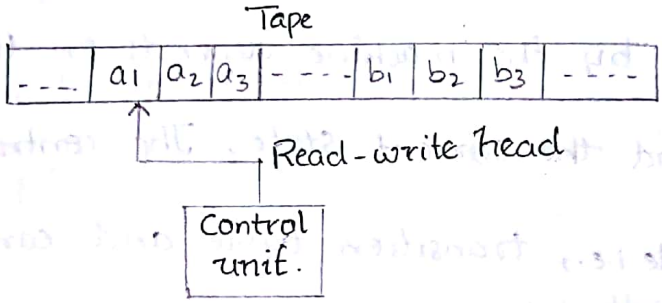
1. With a neat diagram, explain the working a basic Turing machine.

A.



Turing machine:

The Turing machine is a finite automaton connected to read-write head with the following components:

- Tape
- Read-write head
- Control unit.

Tape: Tape is used to store the information and tape is divided into the cells. Each cell can store the information of only one symbol. The string to be scanned will be stored from the leftmost position on the tape. The string to be scanned should end with blanks. The tape is assumed to be infinite both on left side and right side of the string.

Read-write head: The read and write head can read a ~~table~~ symbol from where it is pointing to and it can write into the tape to where it points to.

Control unit: The reading from tape or writing into the tape is determined by the control unit. The different moves performed by the machine depends on the current scanned symbol and the current state. The control unit consults action table i.e., transition table and carry out the tasks.

The read-write head can move either towards left or right i.e., movement can be on both the directions. The various actions performed by the machine are:

- Change of state from one state to another state
- The symbol pointing to by the read-write head can be replaced by another symbol.
- The read-write head may move either towards left or right.

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

2. Obtain a Turing machine to accept the language

$L = \{0^n 1^n \mid n \geq 1\}$

A. $L = \{0^n 1^n \mid n \geq 1\}$

Language L accepted by TM should have 'n' no.of 0's followed by 'n' no.of 1's.

$$L = \{01, 0011, 000111, \ldots\}$$

→ Let q_0 be the start state and let the read-write head points to the first symbol of the string to be scanned.

. Replace the leftmost 0 by X and change the state to q_1 and then move read-write head towards right.

. Search for the leftmost 1 and replaces it by the symbol Y and move towards left.

The above two steps can be repeated.

Step 1: In state q_0, replace 0 by X, change state to q_1 and move pointer towards right.

$$\delta(q_0, 0) = (q_1, X, R)$$

Step 2: In state q_1, obtain leftmost 1 and replace it by Y, and move towards left. When pointer moves towards 1, symbols encountered may be 0 and Y, irrespective of that, replace 0 by 0, Y by Y and remain in same state and move pointer towards right.

$$\delta(q_1, 0) = (q_1, 0, R)$$
$$\delta(q_1, Y) = (q_1, Y, R)$$

Step 3: In state q_1, if input symbol to be scanned is a 1, then replace 1 by Y, change state to q_2 and move left.

$$\delta(q_1, 1) = (q_2, Y, L)$$

Step 4: To obtain leftmost zero, we need to obtain rightmost x first, during this process, we may encounter Y's and 0's, replace Y by Y and 0 by 0, and remain in state q_2.

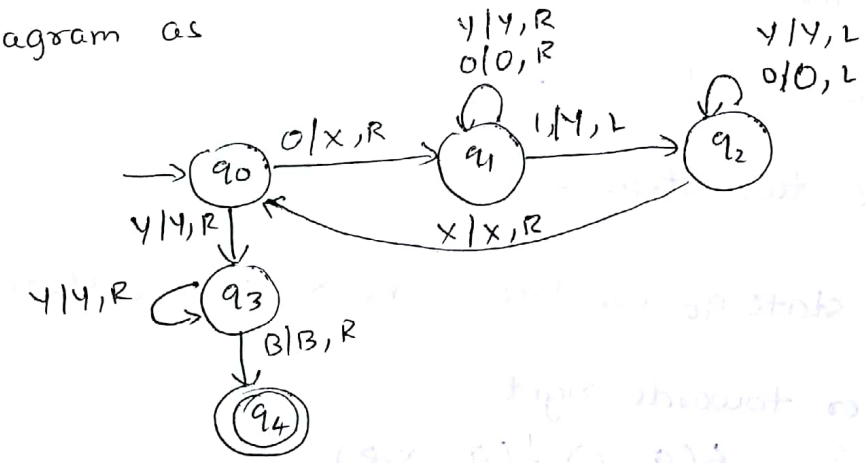
$$\delta(q_2, Y) = (q_2, Y, L)$$

$$\delta(q_2, 0) = (q_2, 0, L)$$

Step 5: Now to obtain leftmost 0, replace x by x, change state frome q_2 to q_0 and move right.

$$\delta(q_2, x) = (q_0, x, R)$$

Repeating Steps 1 to 5 we get configuration of transition diagram as



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

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

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

$\Gamma = \{0, 1, x, y, B\}$

$q_0 = \{q_0\}$ start state

B = Blank character

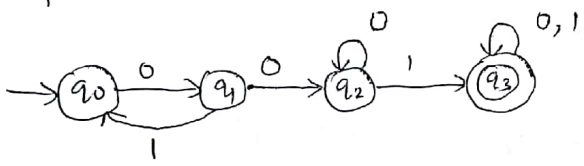
$F = \{q_4\}$ final state

δ	Tape symbols (Γ)				
States	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

3. Design a Turing Machine to accept following language $L = \{ w \mid w \in (0+1)^* \}$ containing the substring 001.

A. $L = \{ w \mid w \in (0+1)^* \}$ containing the substring 001

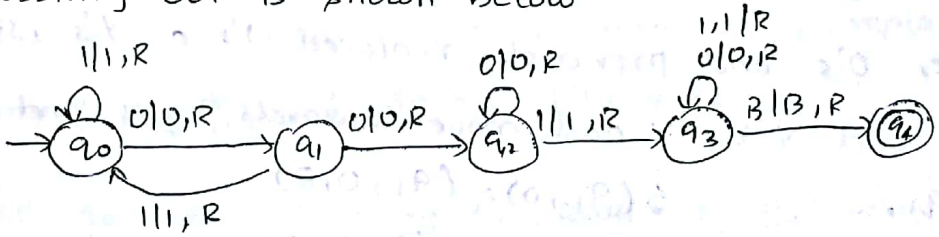
DFA for this is :



→ As 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, whichever state the DFA was in, TM also enters into same states on same input symbols, replacing 0 by 0 and 1 by 1 and read-write head moves towards right.

→ So TM to recognize language of 0's and 1's having a substring 001 is shown below



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

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

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

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

q_0 = start state

B = Blank character

$F = \{q_4\}$ final state.

States	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
$*q_4$	-	-	-

4. Obtain a Turing machine to recognize the language

$L = \{0^n 1^n 2^n \mid n \geq 1\}$

A. $L = \{0^n 1^n 2^n \mid n \geq 1\}$, so language should consists of 'n' no. of 0's followed by 'n' no. of 1's followed by 'n' no. of 2's.

Let us suppose take q_0 as start state.

→ In state q_0, if next scanned symbol is 0, replace it by X and change state to q_1 and move right.

$$\delta(q_0, 0) = (q_1, X, R)$$

→ In state q_1, search for leftmost 1 in this process we may encounter 0's and previously replaced 1's as Y's, so replace 0 by 0 and Y by Y. and move towards right and remain in state q_1.

$$\delta(q_1, 0) = (q_1, 0, R)$$

$$\delta(q_1, Y) = (q_1, Y, R)$$

→ In state q_1, on encountering 1 change to state q_2 and replace 1 by Y and move right.

$$\delta(q_1, 1) = (q_2, Y, R)$$

→ In state q_2, search for leftmost 2, in this process we may encounter 1 or Z (previously replaced 2's), so replace 1 by 1, Z by Z and move right remaining in same state.

$$\delta(q_2, 1) = (q_2, 1, R)$$
$$\delta(q_2, Z) = (q_2, Z, R)$$

→ In state q_2, on encountering 2 change to state q_3 and replace 2 by Z and move left

$$\delta(q_2, 2) = (q_3, Z, L)$$

→ In state q_3, we have to search for rightmost X to get leftmost 0. During this process, we may encounter Z, 1, 0, Y, so replace X by Y, 0 by 0, 1 by 1, Z by Z and move left.

$$\delta(q_3, Z) = (q_3, Z, L)$$
$$\delta(q_3, 1) = (q_3, 1, L)$$
$$\delta(q_3, Y) = (q_3, Y, L)$$
$$\delta(q_3, 0) = (q_3, 0, L)$$

On encountering X, replace X by X, change state to q_0 and move right. $\delta(q_3, X) = (q_0, X, R)$

In state q_0, if input is Y, it means that there are no 0's. If there are no 0's we should see that there are no 1's also. For this to happen, change state to q_4, replace Y by Y and move right $\delta(q_0, Y) = (q_4, Y, R)$

In q_4, replace Y by Y, remain in q_4 and move right.

$$\delta(q_4, Y) = (q_4, Y, R)$$

In q_4, if we encounter Z, it means that there are no 0's and 1's, so we should see for 2's, and only Z's should be present. Scanning the 1st Z, change state to q_5, replace Z by Z and move right. $\delta(q_4, Z) = (q_5, Z, R)$

In state q_5, there should be only Z's so scan all Z's and replace by Z and if B is encountered move to state q_6 the final state $\delta(q_5, Z) = (q_5, Z, R)$

$\delta(q_5, B) = (q_6, B, R)$

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

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

$\Sigma = \{0, 1, 2\}$ $\Gamma = \{0, 1, 2, X, Y, Z, B\}$

$q_0 = \{q_0\}$ – start state B – Blank character. $F = \{q_6\}$ – final state.

	Γ						
States	0	1	2	Z	Y	X	B
q_0	q_1, X, R	–	–	–	q_4, Y, R	–	–
q_1	$q_1, 0, R$	q_2, Y, R	–	–	q_1, Y, R	–	–
q_2	–	$q_2, 1, R$	q_3, Z, L	q_2, Z, R	–	–	–
q_3	$q_3, 0, L$	$q_3, 1, L$	–	q_3, Z, R	q_3, Y, L	q_0, X, R	–
q_4	–	–	–	q_5, Z, R	q_4, Y, R	~~q_0, X, R~~	–
q_5	–	–	–	q_5, Z, R	–	–	q_6, B, R
q_6	–	–	–	–	–	–	–

5. Briefly explain the techniques for TM construction.

A. The various techniques used for constructing a TM are shown below:

1. Turing machine with stationary head: In Standard Turing machine δ is defined as

$\delta(q, a) = (p, y, D)$

where D stands for direction. So, D can be left or right denoted by L or R. Here, after consuming input symbol a, the R/w head moves either towards left or right. An option such that the R/w head remains in same cell for some input symbols can be implemented.

$$\delta(q, a) = (p, y, S)$$

The TM in state q, after consuming 'a', replace 'a' by 'y' and change state from q to p, without updating R/w head either towards left or right.

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

Q is set of finite states, Σ is set of input alphabets, Γ is set of tape symbols, δ is transition function from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R, S\}$ indicating the TM may move towards left or right or stay in same position, q_0 is start state, B - blank character, $F \subseteq Q$ is set of final states.

2) Storage in the state: In TM where we store in a state, that state becomes a pair (q, a) where q is the state and a is a tape symbol stored in the state (q, a). So new set of states is given by: $Q \times \Gamma$.

3) Multiple track Turing Machine: In a standard TM only one tape was used to store the data. In multiple track TM, a single tape is assumed to be divided into a number of tracks. If a single tape is divided into K tracks, then the tape alphabets consist of K-tuples of tape symbols. The working remain same as that of standard TM.

4) Subroutines: We know that subroutines are used in

programming languages whenever a task has to be done repeatedly. The same facility can be used in TM and complicated TM's can be built using subroutines. A TM subroutine is a set of states that performs some pre-defined task. It has a start state and a state without any moves. This state which has no moves serves as the return state and passes the control to state which calls the subroutine.

6. Define the following terms with an example with respect to TM

i) Transition table ii) Instantaneous description iii) Transition diagram.

A. Transition Table: The transitions of TM represented using a table is called transition table of a TM

Ex:

δ	Γ				
States	a	b	X	Y	B
q_0	q_1, X, R			q_3, Y, R	
q_1	q_1, a, R	q_2, Y, L		q_1, Y, R	
q_2	q_2, a, L		q_0, X, R	q_2, Y, L	
q_3				q_3, Y, R	q_4, B, R
q_4					

For each state q, there is a corresponding entry for the symbol in Γ.

Undefined entries in table indicate that there are no transitions defined.

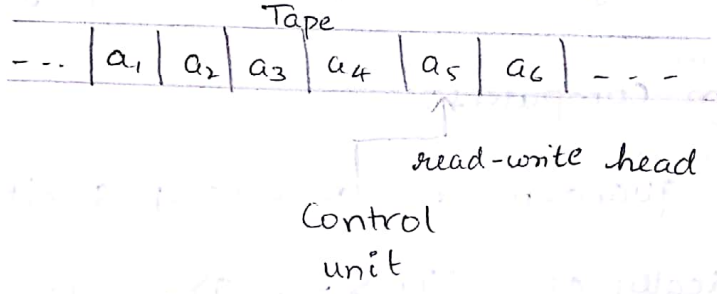
$\delta : Q \times \Gamma$ to $(Q \times \Gamma \times \{L, R\})$

When there is a transition to dead state, machine halts and input string is rejected by the machine.

Instantaneous description (ID):

An ID of TM is defined as a string : $\alpha q \beta$ where

- q is current state of TM
- given string is divided into two strings α and β such that given string is obtained by concatenating α and β.
- Initial ID is denoted by $q\alpha\beta$
- Final ID is denoted by $\alpha\beta q B$



ID's are:

$a_1a_2a_3a_4$ – towards left of state q_2 is the left sequence.

a_5a_6 – towards right of state q_2 is right sequence.

q_2 – current state.

a_5 – next symbol to be scanned.

Transition diagram: Transition diagram for a TM

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

is defined as a graph with circles and arrows, arc with labels and two concentric circles where

- each state is called a vertex
- each edge is labelled with $(a, X, L|R)$
- Transition from one state to another state is indicated by a directed edge.
- Start state is a state which has an arrow not originating from any node, and entering into state.
- Final states which are in F are represented by double circles.

Ex:

7. With example, explain quantum computation.

A. In 1982, Richard Feynmann, a nobel laurate in physics suggested that scientist should start thinking of building computers based on the principles of quantum mechanics. Quantum computation based on principles of quantum mechanics will provide tools to fill up gulf between small and relatively complex systems in physics.

Quantum computers:

A quantum bit or simply qubit can be described mathematically as $|\psi\rangle = \alpha|0\rangle + \beta|0\rangle$

Two possible states for a qubit are state $|0\rangle$ and $|1\rangle$. A qubit can be in infinite number of states other than $|0\rangle$ and $|1\rangle$. It can be in any state $|\psi\rangle = \alpha|0\rangle + \beta|0\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. Here 0 and 1 are called computational basis states and $|\psi\rangle$ is called a superposition. We can call $|\psi\rangle = \alpha|0\rangle + \beta|0\rangle$ a quantum state. We either get state $|0\rangle$ with probability of $|\alpha|^2$ or state $|1\rangle$ with probability of $|\beta|^2$.

For example $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$ and $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$ with $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

Classical NOT gate interchanges 0 and 1, but qubit NOT gate, $\alpha|0\rangle + \beta|1\rangle$ is changed to $\alpha|1\rangle + \beta|0\rangle$.

It is built from quantum circuits.

Church Turing Hypothesis :

Various formal models of computations such as Recursive functions and post systems were established by three prominent persons A. Church, S.C. Kleene and E. Post.

A function is called primitive recursive if and only if it can be constructed from the basic functions by successive composition and primitive recursion. A Post System is similar to unrestricted grammar consisting of an alphabet and some production rules by which successive strings can be derived.

Computational model looks quite different from post and recursive functions, this observation was formalized in Church's thesis stated as

" Any Effective computation or any algorithmic procedure that can be carried out by a human being or a team of human beings or a computer, can be carried out by some Turing Machine.

This thesis predicts that if is unable to construct models of computation more powerful than existing ones.

Since there is no precise definition for "effective computation" or "algorithmic procedure", Church thesis is not a mathematically precise statement.

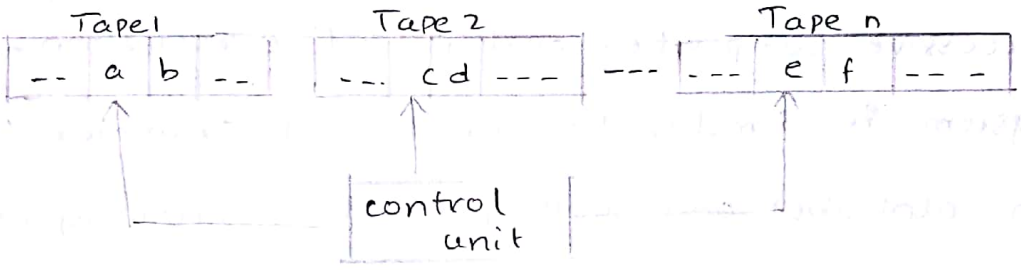
8. Write a short note on the following

a. Variants of Turing machine

b. linear bounded automata.

A. a. Variants of Turing machine are

1) Multi-tape TM:



A multi-tape TM is nothing but a standard TM with more number of tapes. Each tape is controlled independently with independent read-write head. It has components such as finite control, multiple R/w heads, and each tape is divided into cells. R/w heads of each tape can move left or right accordingly.

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

where Q is set of finite states

Σ input symbols set

Γ set of tape symbols, δ is transition, q_0 is start state

B is blank character and F is final state.

2) Non-deterministic Turing Machine:

The non-deterministic TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where Q is set of finite states, Σ is set of input symbols, Γ is set of tape symbols, δ is transition, q_0 is start state, B is Blank character, F is final state.

Language accepted by non-deterministic TM is

$$L(M) = \{ w \mid q_0 w \vdash^* \alpha_1 p \alpha_2 \text{ where } w \in \Sigma^*, p \in F \text{ and } \alpha_1, \alpha_2 \in \Gamma^* \}$$

Non-deterministic TM may have many moves that does not lead to a final state. It is no powerful than deterministic TM. Both deterministic and non-deterministic TM are equivalent.

b. Linear bounded Automata (LBA):

With slight alteration in usage of the tape, let us restrict the workspace on tape using two delimiters '[' and ']'. The given string has to be enclosed between these 2 delimiters. Longer the string, longer the workspace. This leads to another class of machine called linear bounded automata (LBA). So, LBA is a TM, which is bounded based on length of input string. LBA is a TM

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

where Q is set of finite states, Σ is set of input symbols, Γ is tape-symbols set, δ is transition $\delta : Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$

q_0 is start state, B is blank character, F is final state

Σ contains two special symbols here '[' and ']'.

9. Prove that halting problem of Turing machine is undecidable.

A. Let L_{HLT} is decidable and get a contradiction. Let M_1

is a TM such that $L(M_1) = L_{HLT}$ and let M_1 halt eventually halts on all (M,w). Let us construct TM M_2 as follows

1) M_2 has (M,w) as input.
2) M_1 acts on (M,w)
3) M_1 rejects (M,w) then M_2 rejects (M,w)
4) M_1 accepts (M,w), simulate TM M on input string w until M halts
5) If M has accepted w, M_2 accepts (M,w) otherwise rejects.

* When M_1 accepts (M,w), TM M halts on w
* In first case, M_2 accepts (M,w)
* In second case, M_2 rejects (M,w)

So, it follows from definition of M_2 that TM M_2 halts eventually and hence, $L(M_2) = \{(M,w) : \text{TM accepts } w\}$ which is undecidable. This is a contradiction. So, the TM M halts on input w is undecidable.

10. Define class P and NP with an example wrt TM.

A. Class P: Let $L(M)$ is the language accepted by TM with time complexity $T(n)$. Language L is in class P if there exists some polynomial $p(n)$ such that $L = L(M)$ for some deterministic TM M with time complexity $T(n)$.

Ex: $L = \{a^n b^n \mid n \geq 1\}$ has time complexity of $O(n^2)$.

Step 1: Every leftmost a is replaced by x and every leftmost b is replaced by y. So total no. of moves = 2n.

Step 2: We have to repeat step 1 n times till all a's are replaced.

Step 3: total no. of moves = $2n * n = 2n^2 \approx n^2$

Time complexity = $O(n^2)$

Class NP: Let $L(M)$ is the language ~~of~~ accepted by TM with time complexity $T(n)$. Language L is in class NP if there exists some polynomial $p(n)$ such that $L = L(M)$ for some non-deterministic TM M with Time complexity $T(n)$.

11. Write a short note on

a. Post correspondence problem

b. Decidability.

A. a. Post correspondence problem:

It can be stated as follows. Given two sequences of n strings on some alphabet Σ say:

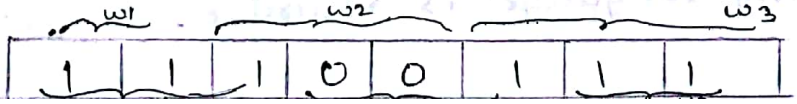
$A = w_1, w_2, w_3, \ldots w_n$ and $B = v_1, v_2, \ldots, v_n$ we say that there exists a post correspondence solution for pair (A,B) if there exists a non-empty sequence of integers $i, j, \ldots, k$ such that $w_i w_j \ldots w_k = v_i v_j \ldots v_k$.

This is to devise an algorithm that will inform us, for any (A,B) whether or not there exists a PC solution.

Ex: Let $\Sigma = \{0,1\}$, Let $A = w_1, w_2, w_3$

such that $w_1 = 11$ $w_2 = 100$ $w_3 = 111$

B is $v_1, v_2 v_3$ as $v_1 = 111$ $v_2 = 001$ $v_3 = 11$



If we take $w_1=00, w_2=001, w_3=1000$

$v_1=0, v_2=11, v_3=011$ there cannot be any PC solution simply because any string composed of elements of A will be longer than corresponding string from B.

b. Decidability:

As an algorithm terminates eventually, TM also terminates. TM halts in following 2 situations:

* When a TM reaches a final state
* When a TM reaches a state q, when scanned input symbol is a and if transition $\delta(q,a)$ is not defined.

But, there are some TMs that never halts on some inputs.

A language L is recursively enumerable if and only if there exists a TM M such that $L=T(M)$ where $T(M)$ is the language accepted by TM.

A Language L is recursive if and only if there exists a TM if $w \in L$ then w is accepted by TM on reaching final state or $w \notin L$ then w is rejected by TM without reaching final state. and machine halts.

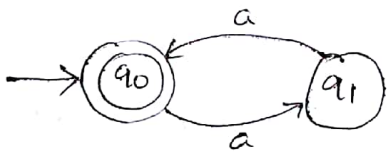
All such problems defined above has ~~yes~~ accept or reject as output called membership problem. All problems with answers yes/no, accept/reject, finite/infinite are called decision problems.

A language is decidable if and only if the corresponding language is recursive.

12. Design a TM to accept even no.of a's. Show the sequence of moves made by TM for the strings aa and aaa.

A. Language accepted by TM is even no.of a's.

Since language is regular, first construct a DFA as shown below



Initially TM will be in start state q_0. Here main task is to identify even no.of a's which can be done as shown below:

q_0 : Input symbol may be B or a

- If input symbol is B, then string has even no.of a's and machine enter to final state q_f, replacing B by B. and moving Rlw head towards right

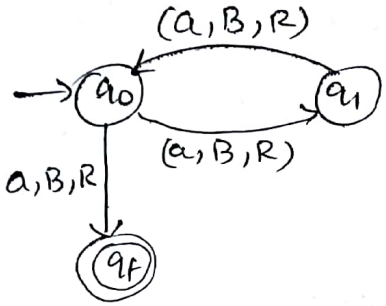
$$\delta(q_0, B) = (q_f, B, R)$$

- If input symbol is a, change state q_1 replacing a by B and move Rlw head towards right.

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

q_1 : If we accept one a, we have even no.of a's and go back to q_0, replacing a by B and moving Rlw head towards right.

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



δ	Γ	
States	a	B
q_0	q_1, B, R	q_f, B, R
q_1	q_0, B, R	–
$*q_f$	–	–

Moves for string aa:

$q_0aaB \vdash Bq_1aB \vdash BBq_0B \vdash BBBBq_f$ (accept)

aa is accepted by Turing machine.

Moves for string aaa:

$q_0aaaB \vdash Bq_1aaB \vdash BBq_0aB \vdash BBBq_1B$ (reject)

Since q_1 on B transition is not defined, aaa is rejected.