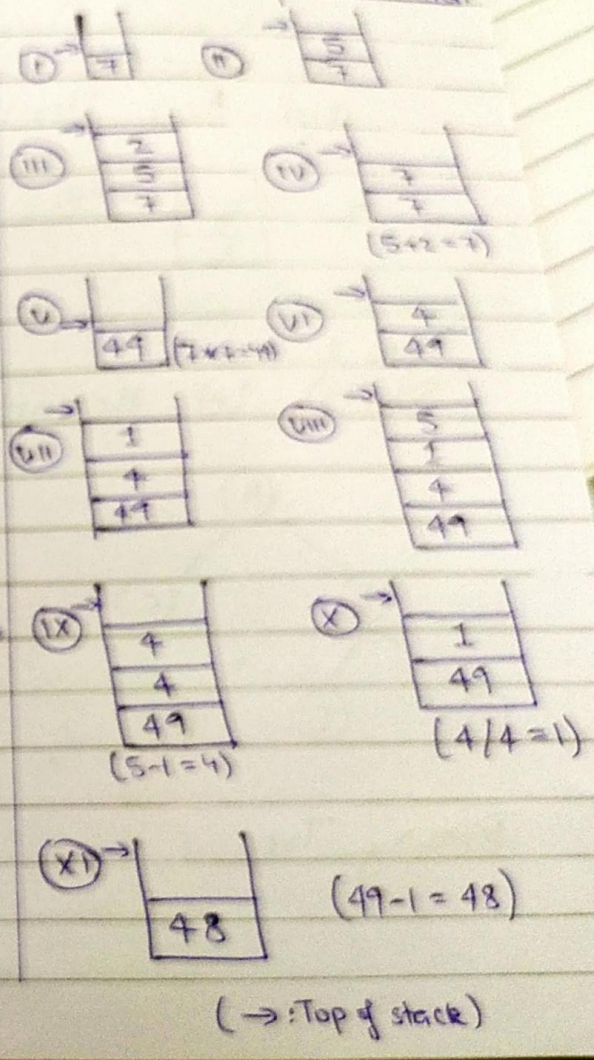
Ques 6

Sol (b) Evaluate : 7 5 2 + * 4 1 5 - / -

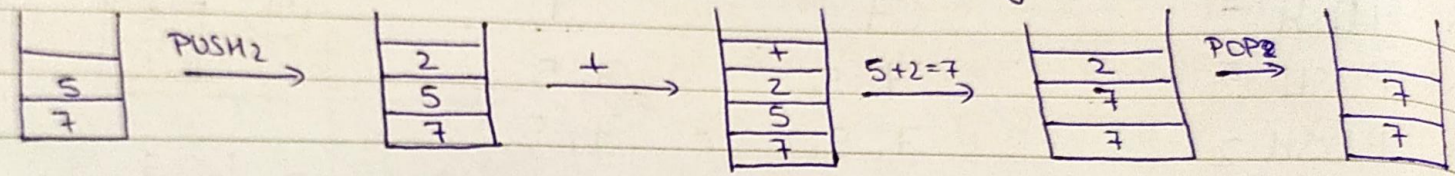
	Operation	Evaluation	Stack
(i)	PUSH 7	7	7
(ii)	PUSH 5	7, 5	7, 5
(iii)	PUSH 2	7, 5, 2	7, 5, 2
(iv)	+	5+2=7	7, 7
		(Add and POP)	
(v)	*	7*7=49	49
		(multiply and POP)	
(vi)	PUSH 4	49, 4	49, 4
(vii)	PUSH 1	49, 4, 1	49, 4, 1
(viii)	PUSH 5	49, 4, 1, 5	49, 4, 1, 5
(ix)	-	5-1=4	49, 4, 4
		(subtract and POP)	
(x)	/	4/4=1	49, 1
		(divide and POP)	
(xi)	-	49-1=48	48
		(subtract and POP)	

Stack Step Presentation



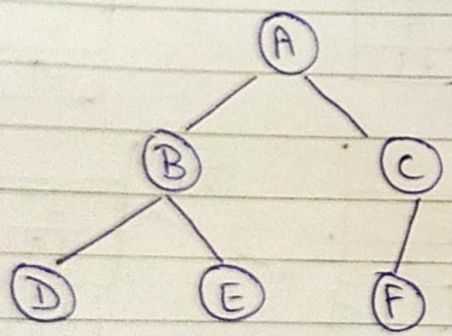
After evaluating the given postfix expression using stack we get 48 as the answer.

In steps (IV), (V), (IX), (X) and (XI), the operator is evaluated and then item is POPed from Top of Stack.



Ques 7

Sol (b) Let the nodes be A, B, C, D and E, A being root node



A → root node

B → parent node for D and E
→ successor for A

C → parent node for F
→ successor for A

D, E, F → leaf / terminal node having no children.

Binary Tree

→ It can be implemented using array and linked list both.

→ In memory they are stored in non-contiguous allocations. They are stored in 

post order (or) preorder traversal.

→ Leaf nodes are the terminal nodes having no children.

Level → Each tree is leveled i.e. root node is 0 then its children 1 and so on.

Section B

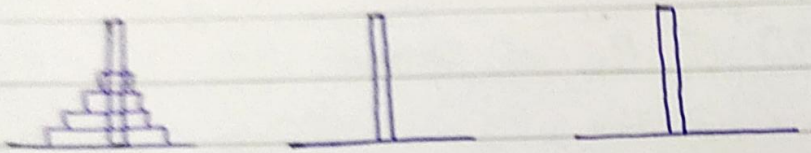
Ques 2

Sol (a) Recursive → function that directly / indirectly calls itself until a specific condition is met.

Tower of Hanoi is a fun task to place all disks from source to destination using spare. It has 2 rules:

(i) Only 1 disk at a time can be moved.

(ii) No bigger disk can be placed on top of smaller disk.



Algorithm

(i) Give no to all disks (1, 2, ...n)

(ii) Move disk in ascending order (1 to spare, 2 to destination)

(iii) Move disk from spare to destination (1 from spare to destinati

(iv) Repeating step (ii) & (iii) until all disks are moved from source to destination.

(v) End of program.

Ques 3

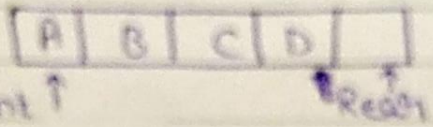
Sol (b) Role of Front and Rear

→ Queue is a linear data structure which follows LIFO. (Last In First Out)

→ Front is the starting of queue which removes elements

→ Rear is responsible for addition of elements in back of queue.

Eg



Double-ended queue (DEQUE)

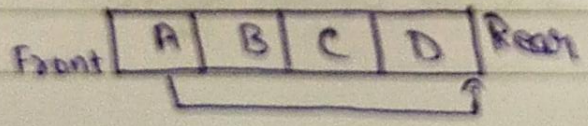
→ It is a queue in which addition or removal of elements can be done from either ends. It is of two types-

① Input restricted: Elements can be added from only one side.

② Output restricted: Element can be removed from only one side

Circular Queue

→ When the front is connected with rear to ease addition and deletion of elements which consumes less memory.



Ques 4

Sol: (a) In order (LNR): B, Z, D, A, C, G, E, H, F

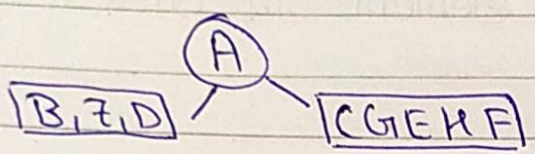
(Left: B, Z, D; root: A; Right: C, G, E, H, F)

Post order (LRN): Z, D, B, G, C, H, F, E, A

From post order → root node: A

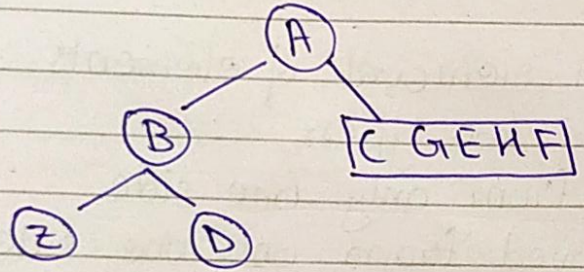
In order follows: Left Node Right

Post order follows: Left Right Node



For left sub tree B, Z, D: root node is B. Z → left

D → right

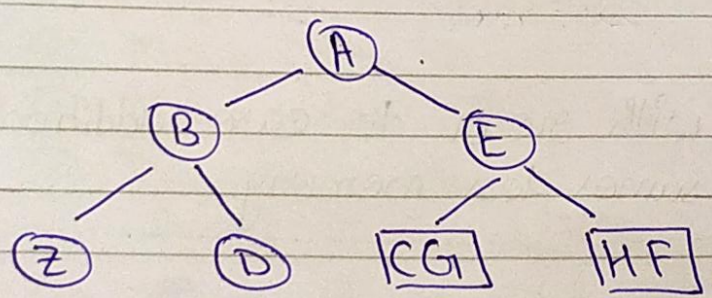


For right subtree C G E H F:

E → root node

C, G → Left

H, F → right

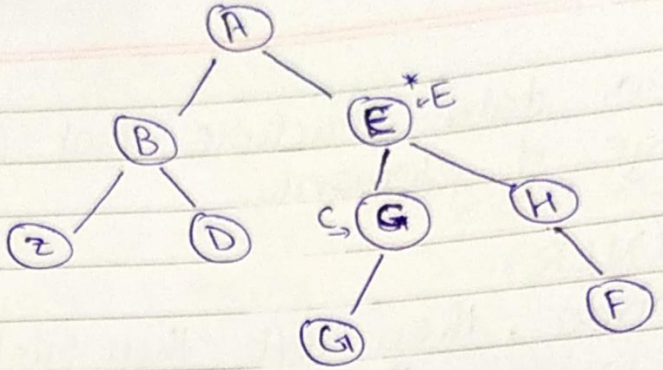


From post order and In order

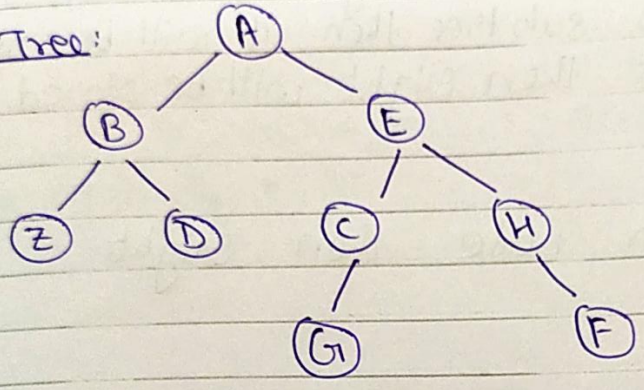
G is left of C

F is right of H

Binary Tree :



Binary Tree :



Ques 5

Sol: (a) There are 3 types of traversal ~~algorithm~~ approaches

(i) Pre order
(ii) In order
(iii) Post order

A tree is non-linear data structure that uses memory allocation for storage of elements.

(i) Pre order traversal (NLR)
It follows root node, then left then right. i.e. first root node is written then its left sub-tree is stored and if subtree has subtree then it will be node for that sub tree and then left then right will be stored.

(ii) In order traversal (LNR)
It follows left subtree then node then right subtree.

(iii) Post order traversal (LRN)
It first traverses left subtree then ~~root node~~ right subtree then root node is traversed.

Ques 1

Section A

(a)

Sol Condition for circular queue : FRONT = 0
to check if queue is empty REAR = -1

(b)

Sol: Tail recursion is when there is no code after recursion to execute is called tail recursion. i.e. recursion is done at the end of code.

(c)

Sol: Stacks are used for performing recursions. i.e. calling a function again and again until a condition is fulfilled.

(d)

Sol: Priority queue is assigning priority to specific elements which will be executed first in a queue.
Advantage → Used in computers (O.S) for performing operations.

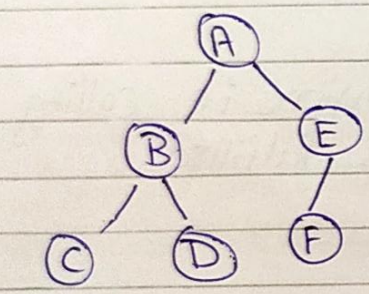
(e) In Tower of Hanoi, total no. of moves $\Rightarrow 2^n - 1$
here $n = 10$,

No. of moves $= 2^{10} - 1$
$= 1024 - 1$
$= 1023$ moves.

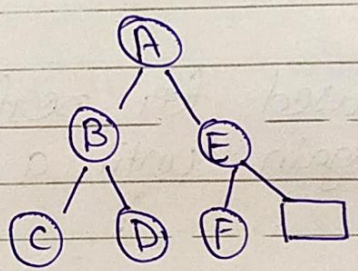
(g) Strict Binary Tree: Tree in which elements are as left as possible

Complete Binary Tree: Tree having all elements i.e. nodes for free space to make it complete.

eg -



Strict Binary Tree



Complete Binary Tree

(h) Binary Search Trees are easy to traverse and can be stored easily as compared to binary trees. They are traversed by Inorder, Preorder and Postorder.

(i) Applications of tree -
(a) Used for traversal of elements.
(b) Storage of elements.
(c) Easier representation of data.
(d) Evaluation of Infix / Prefix