# DAY-08

+ implementation of dynamic queue (by using a linked list)

enqueue -> add_last()
dequeue -> delete_first()

OR

enqueue -> add_first()
dequeue -> delete_last()

+ applications of queue:

- queue is used in any application where collection/list of elements should works in first in first out manner.
- queue is used to implement advanced data structure algorithms like "bfs" (breadth first search) in tree & graph.
- queue is used to implement kernel data structures like ready queue, job queue, message queue etc...
- queue is used to implement os algorithms like priority cpu sched, fcfs cpu sched algo etc...

==

+ advanced data structures/non-linear data structures:

1. "tree": it is "non-linear", "advanced data structure", which is a collection/list of logically related finite no. of elements, in which there is a first specially designated element referred as a "root" element, and remaning all ele's are connected to the root ele in a hierachical manner, follows parent-child relationship.

- in a tree data structure element is also called as a "node"

* first ele in tree = root node/root ele
* parent node/father
* child node/son
* grand parent/grand father
* grand child/grand son
* siblings/brothers: child nodes of same parent
* degree of a node = no. of child nodes having that node
* degree of a tree = max degree of any node in a given tree
* leaf node/terminal node/external node:
node which is having degree 0
OR node which is not having any child node

* non-leaf node/non-terminal node/internal node:
node which is having non-zero degree
OR node which is having any no. child node

* ancestors: all the nodes which are in the path from root node to that node (including root node/excluding itself).
- root node is an anscestor for all the nodes.

* descendents: all the node which can be accessible from that node
- all the nodes are descedents of root node

* level of a node = level of its parent node + 1
if we assume: level of a root node = 0

* level of a tree = max level of any node in a given tree
* depth of a tree = level of a tree

- in a tree data structure, any node can have any no. of child nodes, and
it can grow at any level.
- if we want to achieve operations on a tree data structure efficiently few restrictions can be applied on it, and hence there are different types of tree:

- "binary tree": it is a tree in which each node can have max 2 no. of child node
OR it is a tree in which each node can have either 0 OR 1 OR 2 no. of child nodes.
OR it is a tree in which each node is having degree either 0 OR 1 OR 2.

set: 0 no. of ele's -> empty set
set: 1 ele ->
set: >1 ele's ->

- binary tree is finite set of elements which has three subsets:
1. root node
2. left subtree (may empty)
3. right subtree (may empty)

- "binary search tree"/BST: it is a binary tree in which left child is always smaller than its parent and right child is always greater than or equal to its parent.

PREORDER : 50 20 10 5 15 45 30 90 85 75 50 100 95 120
INORDER : 5 10 15 20 30 45 50 50 75 85 90 95 100 120
POSTORDER: 5 15 10 30 45 20 50 75 85 95 120 100 90 50