Assignment - 3

1.) Explain following terms with reference to TREE.

→ (i) Node : It is fundamental Part of tree. it contain's data and may have a reference to one or more other nodes

(ii) Root : The top most node in a tree. which has not parent node is called Root.

(iii) Child : A child node is a node directly connected to another node when moving away from the root

(iv) Parent : The node from which other nodes are descended.

(v) Path : A Path is a sequence of node where each adjacent pair is connected by an edge.

(vi) Leaf : A node that has no children.

(vii) Depth : The length of the Path from the root ~~as~~ to the node

(viii) Height : The length of ~~th~~ the longest Path the from the node to leaf.

(ix) Degree : The number of child nodes it h

(x) level : level of a node = depth of node + 1

2.) ~~Explain~~ Define following terms with reference to GRAPH.

→ (i) Finite graph : A graph which has finite set of vertices and edges.

(ii) Infinite graph : A graph which has either set of vertices or set of edges or both infinite.

(iii) Trivial graph : A graph Contain only single vertex and no edges.

(iv) Simple graph : A graph which there is at most one edge between any two distinct vertices, and there are no self loops

(v) multigraph : A graph that have multiple edges between the same pair of vertices

(vi) NULL graph : A graph with no vertices and no edges.

(vii) Complete / Full graph : A simple graph in which there is unique edge between every pair of distinct vertices.

3.) Write an algorithm / code to create Binary tree.

```
→ 1. [Create a tree node]
   (i) newnode = get tree node ( );
   (ii) newNode (key) : key ;
   (iii) new node (Right) : NULL,
   (iv) new node (left) : NULL

  2. [Create Binary tree]
   (i) if (root = NULL) {
           return newnode
   (ii) if (key < root (key)) then
           root (left) ← insert (root (left), key)
        else if (key > root (key)) then
           root (right) ← insert (root (right), key)

  3. [Finished]
        Return root.
```

→ For Traversal (Inorder)

```
if (root != NULL) then
(i) Traversal (root (left))
(ii) write (root (key))
(iii) Traversal (root (Right))
```

4.) What are the different types of TREE traversal mechanisms ?

→ There are three ways of traversing in binary tree

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

(1.) Pre order :-

- Process the root node
- Traverse the left subtree in Preorder
- Traverse the Right subtree in Preorder

• If perticular subtree is empty the traversal is performed by doing nothing.

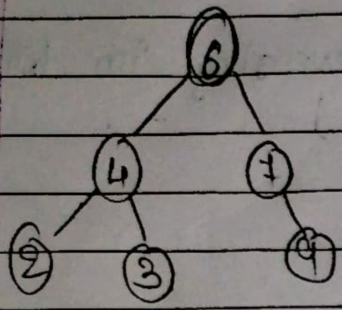
(2.) Inorder Traversal :-

- Traverse the left subtree in Inorder
- Process the root node
- Traverse the Right subtree in Inorder

(3.) Postorder Traversal :

- Traverse the left subtree in Post order
- Po Traverse the Right subtree in Post order
- Process the root node

5.) For following Binary tree illustrate the vario
tree traversal outcome.

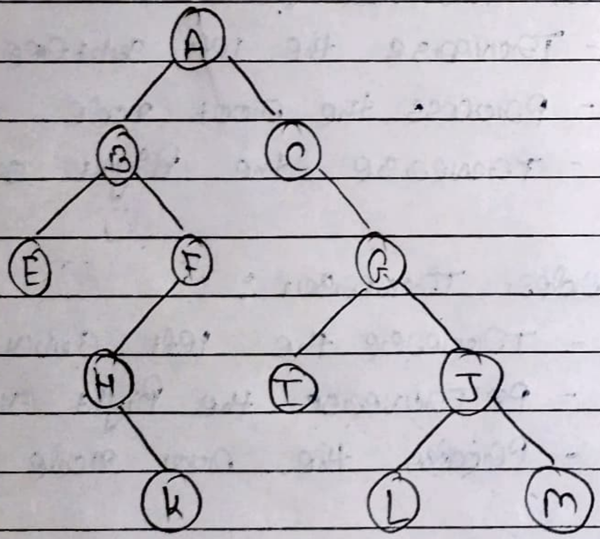


- Pre order : 6 4 2 3 7 9

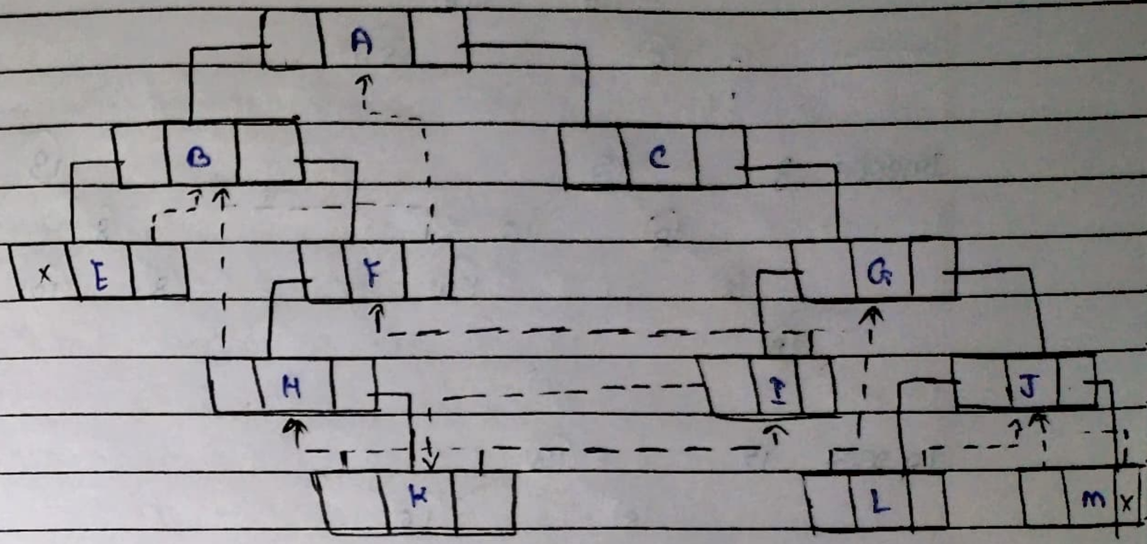
- In order : 2 4 3 6 7 9

- Post order : 2 3 4 9 7 6

6.) Create a Threaded Binary tree for following Binary tree.

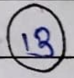


→ Inorder : E · B H K I F A C I G L J M

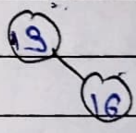


7.) Construct an AVL tree from the given data
13, 16, 10, 8, 54, 3, 12, 11, 9, 59, 20, 15, 21
Show all steps.

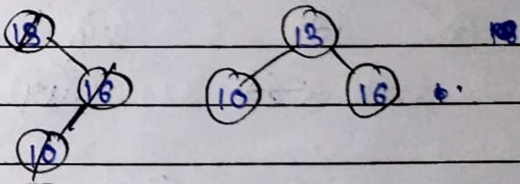
→ Insert : 13



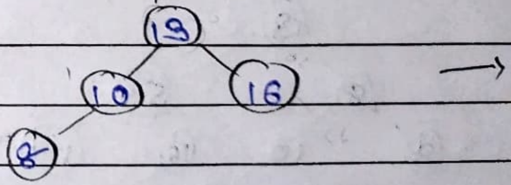
Insert : 16



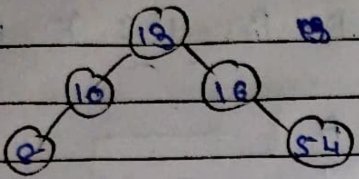
Insert : 10



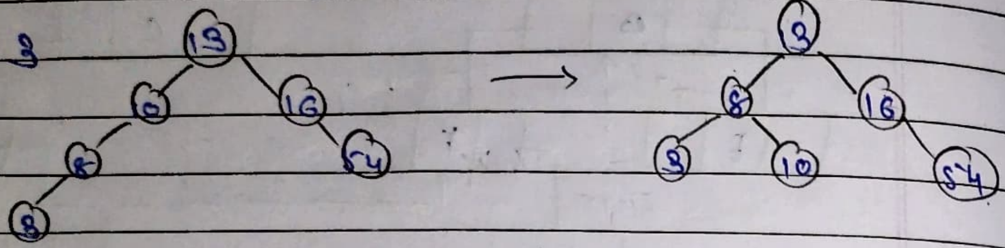
Insert : 8



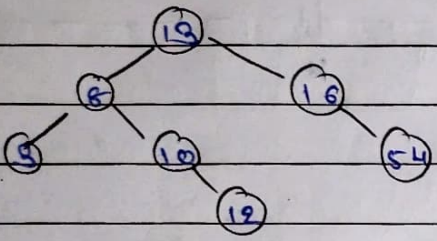
Insert 54



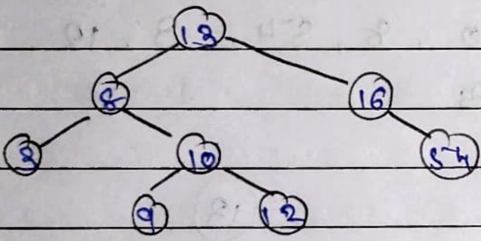
Insert 3



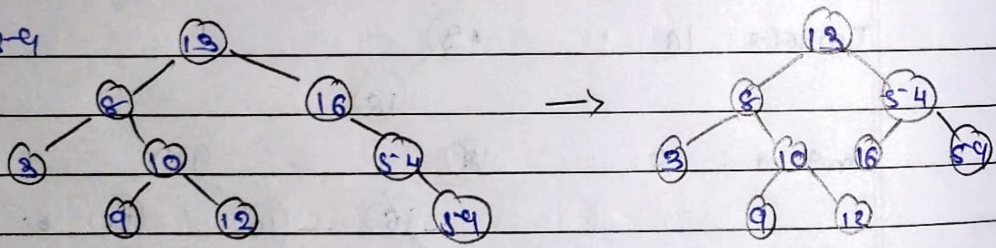
Insert 12



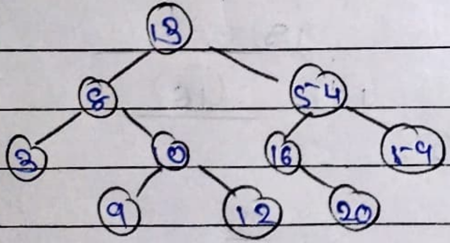
Insert 9



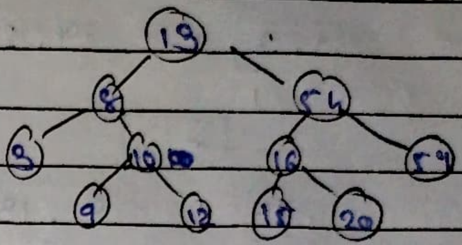
Insert 59



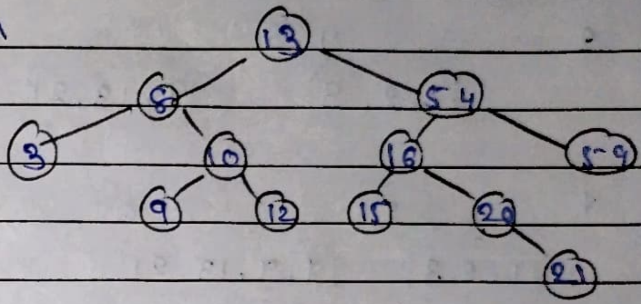
Insert 20



Insert 15 :



Insert 21



8.) Construct a B - Tree (with order m = 4) for given data : 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8

→ order m = 4

max keys = 4-1 = 3

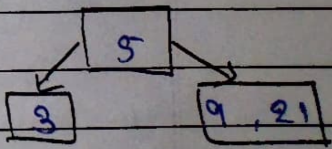
min keys = $\lceil \frac{4}{2} \rceil - 1 = 1$

- Insert 5 $\boxed{5}$

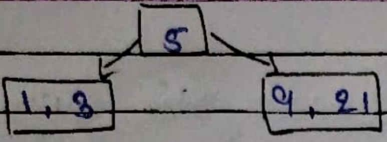
Insert 3 $\boxed{3, 5}$

Insert 21 $\boxed{3, 5, 21}$

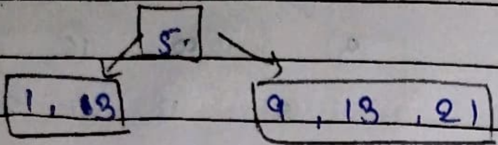
Insert 9 $\boxed{3, 5, 9, 21}$ ⟶



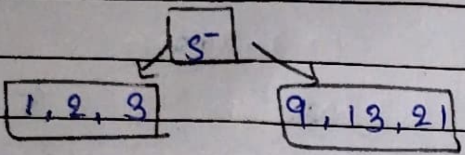
Insert 1



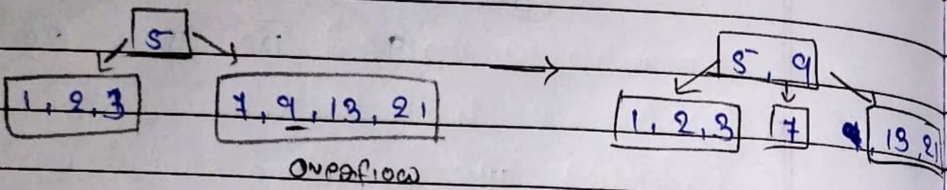
Insert 13



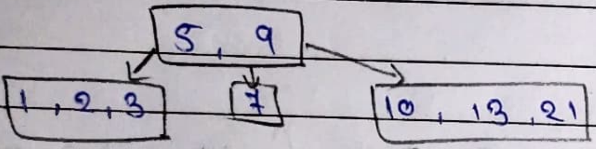
Insert 2



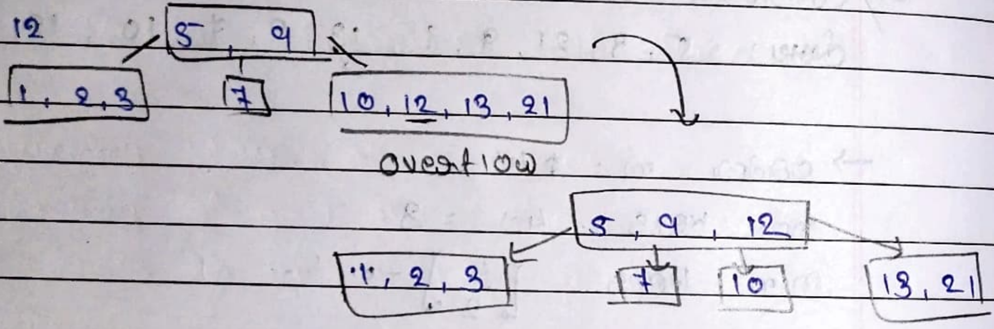
Insert 7



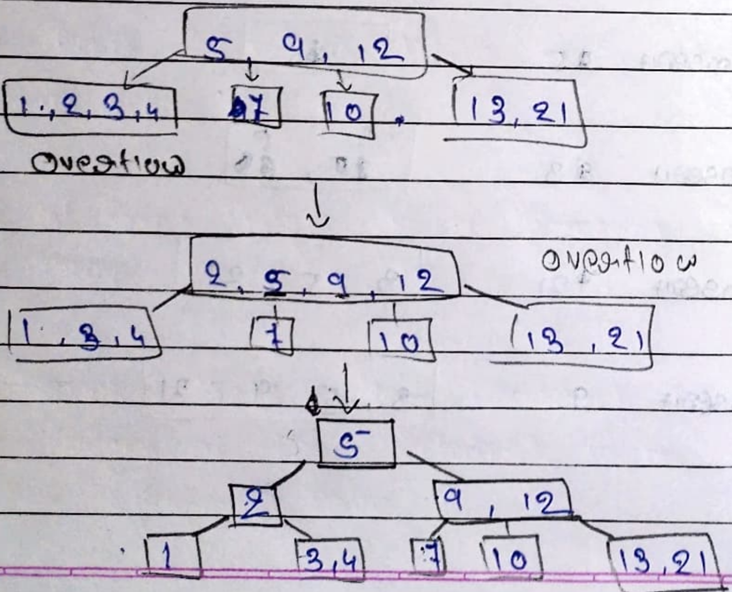
Insert 10



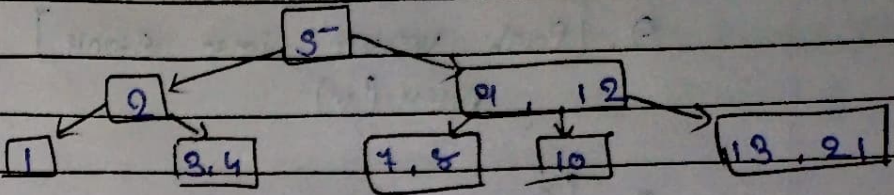
Insert 12



Insert 4



Insert 8 -



10.) Write an algorithm / code for a Breadth First search for a graph.

```
→ 1. Initilize Q

  2. [mark visited node as 1]
        visited (node) ← 1

  3. [Add node in queue]
        insert (node)

  4. [Repeat while Q is not empty]
        while Q is not empty
            node ← Remove ( )
            if visited (node) is 0 then
                visited (node) ← 1
                insert (node)
```

11.) Write an algorithm for Depth First search for a graph.

```
→ 1. [Initilize TOP and visited]
        visited [ ] ← 0
        TOP ← 0
```

```
2. [Push vertex into stack]
        PUSH (v)

3. [Repeat while stack is not empty]
    Repeat step 3    while stack is not empty
          v ← POP ()
          if    visited [v] is 0 then
                visited [v] ← 1
                for all w adjacent to v
                      if visited [w] is 0
                      then PUSH (w)
                End for
          end if
```

~~12.) What is spanning tree?~~

12.) Write general properties of spanning tree

→

~~A~~ A spanning tree of a graph is ~~a~~ an undirecte tree consising of only those edges necessary to Connec all the nodes in the original graph.

- For any pair of nodes there exist only one path between them.

- Insertion of any edge to a spanning tree form a unique cycle

- In DFS search, those edges traversed by the algorith forms the edges of tree, referred to as Depth ~~Seass~~ First spanning tree.

- In BFS search, the spanning tree is formed from those edges traversed during the search producing Breadth first search spanning tree