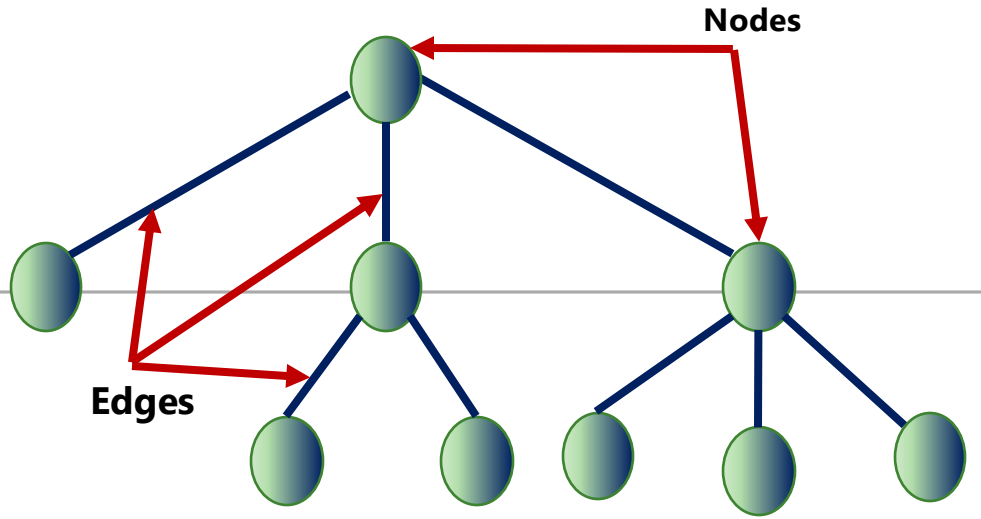

Trees

What is a tree?



1

A tree consist of nodes connected by edges.

2

In a picture of a tree the nodes are represented as circles and the edges as lines connecting the circles.

3

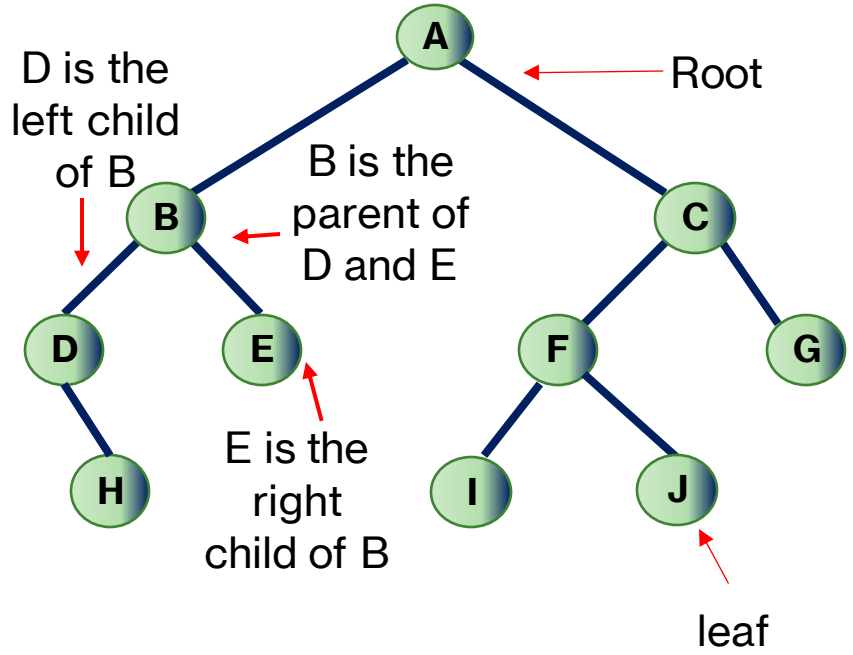
In a tree, the nodes represent the data items and the edges represent the way the nodes are related.

4

A tree with nodes which has maximum of two children is called a **binary tree**.

What is root, parent and children in a tree?

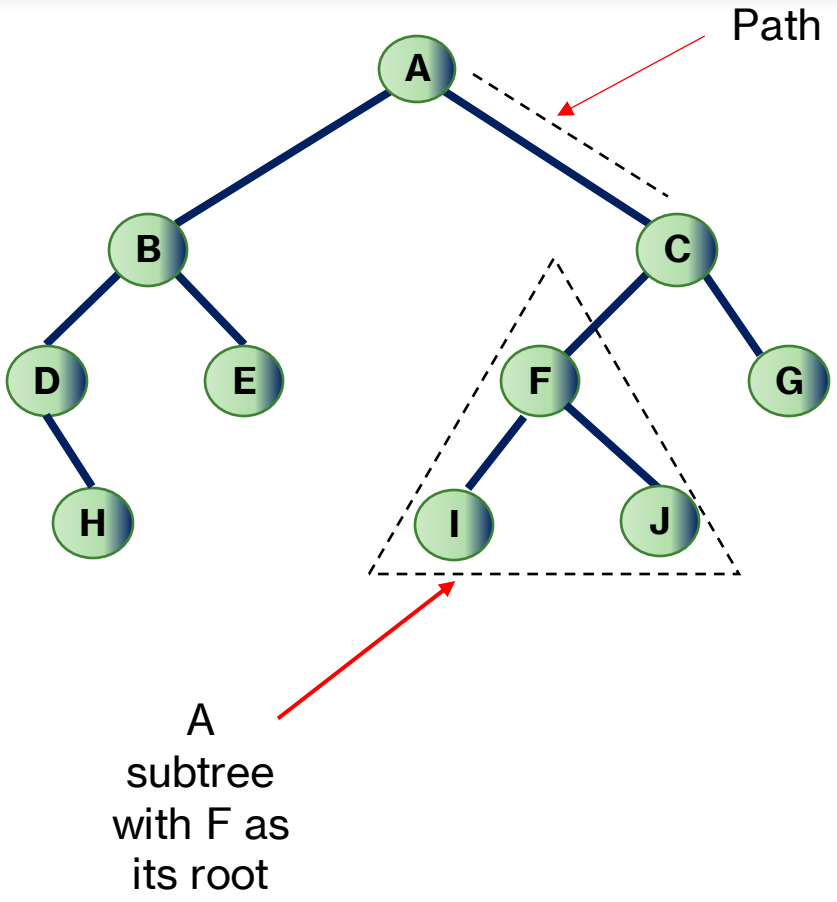
- The node at the top of the tree is called root.
- Any node which has exactly one edge running upwards to other node is called a child
- The two children of each node in a binary tree is called left child and right child.
- Any node which has one or more lines running downwards to other nodes is called a parent
- A node that has no children is called a leaf node or leaf



What is path and subtree in a tree

Sequence of nodes from one node to another along the edges is called a path.

Any node which consist of its children and it's children's children and so on is called a sub tree



Key value of a node

Each node in a tree stores objects containing information.

Therefore one data item is usually designated as a key value.

The key value is used to search for a item or to perform other operations on it.

eg: person object – social security number

car parts object– part number

Binary Search Tree

- is a tree that has at most two children.
- a node’s left child must have a key less than its parent and node’s right child must have a key greater than or equal to its parent.

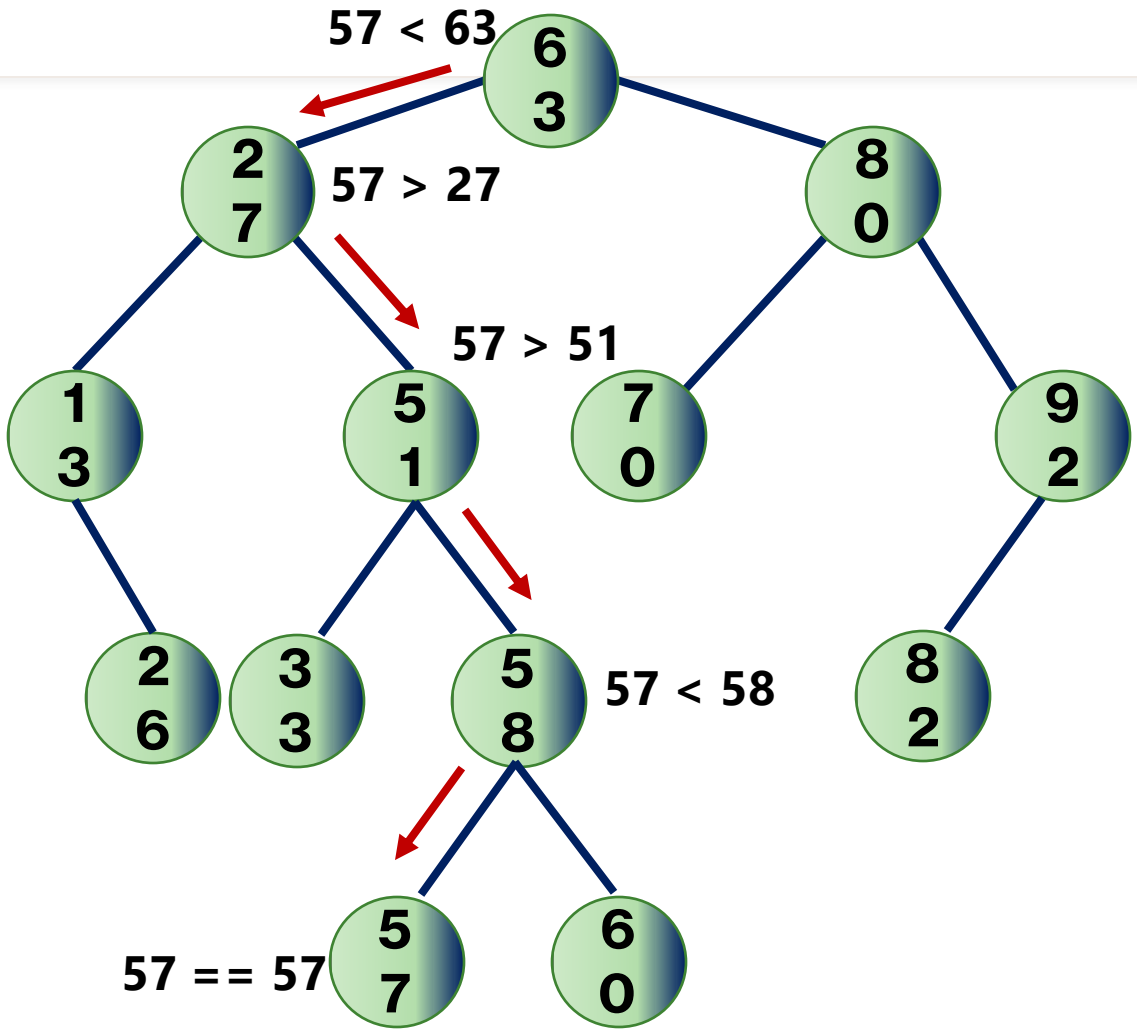
Operations of Binary Search Tree

- There are four main operations perform in a binary search tree
  - Find – find a node with a given key
  - Insert – insert a new node
  - Delete – delete a node
  - Traverse – visit all the nodes

Operations - Find

- Find always start at the root.
- Compare the key value with the value at root.
- If the key value is less, then compare with the value at the left child of root.
- If the key value is higher, then compare with the value at the right child of root
- Repeat this, until the key value is found or reach to a leaf node.

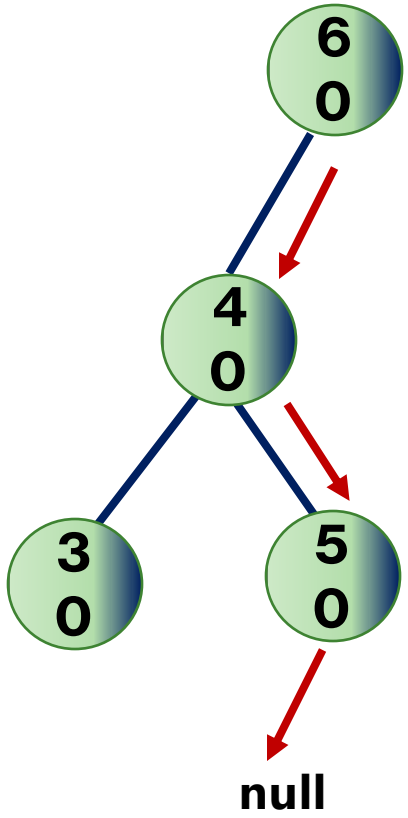
Find value 57



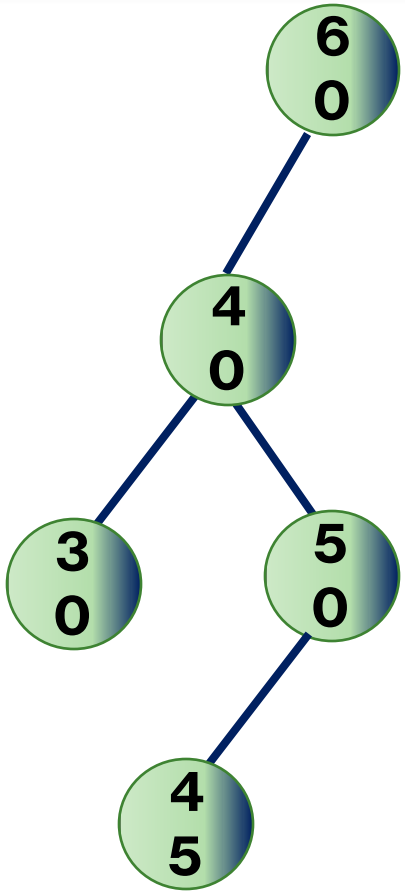
Operations - Insert

- Create a new node.
- Find the place (parent) to insert a new node.
- When the parent is found, the new node is connected as its left or right child, depending on whether the new node's key is less than or greater than that of the parent.

Insert value 45



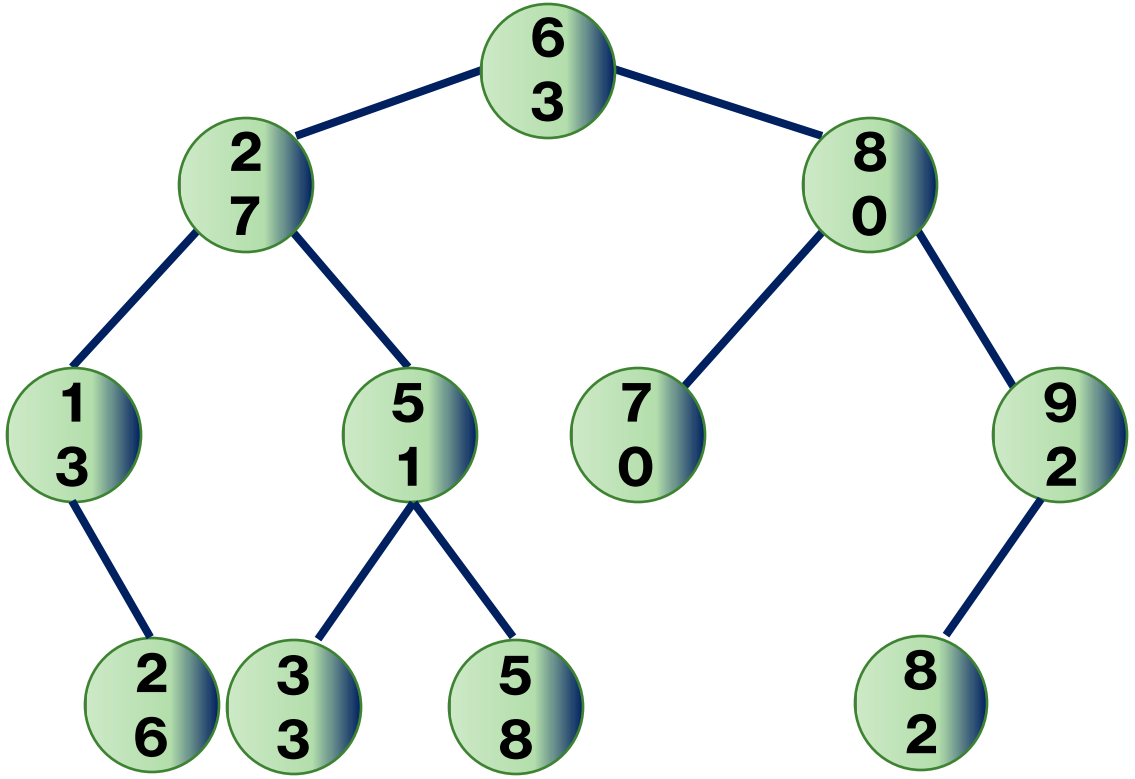
a) Before insertion



b) After insertion

Question 1

- Draw a tree after inserting number 8



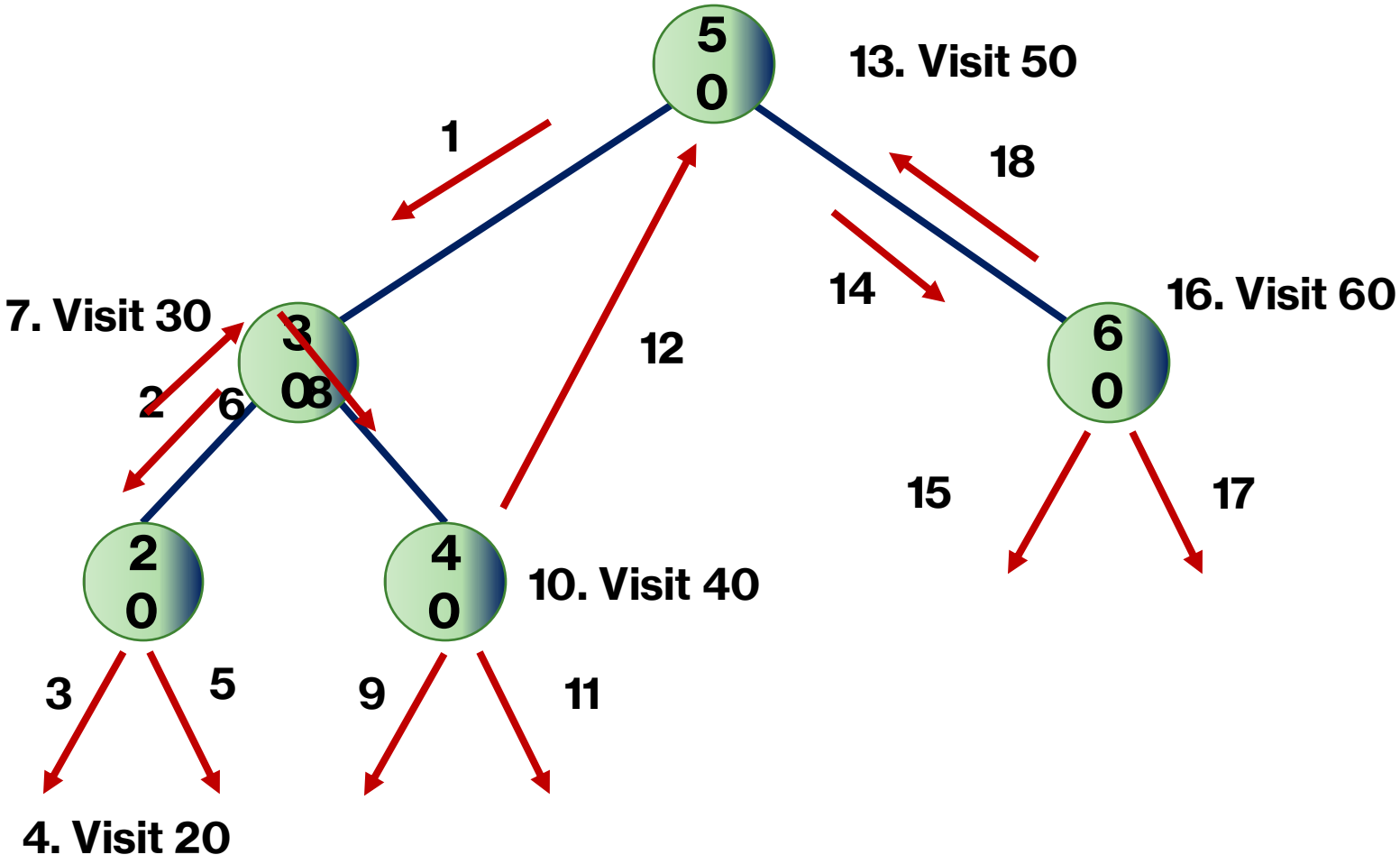
Traversing

- Traverse a tree means to visit all the nodes in some specified order.
- There are three common ways to traverse a tree
  - Pre order
  - In order
  - Post order

Inorder traversing

- Call itself to traverse the node’s left subtree
- Visit the node
- Call itself to traverse the node’s right subtree

Inorder traversing cont...

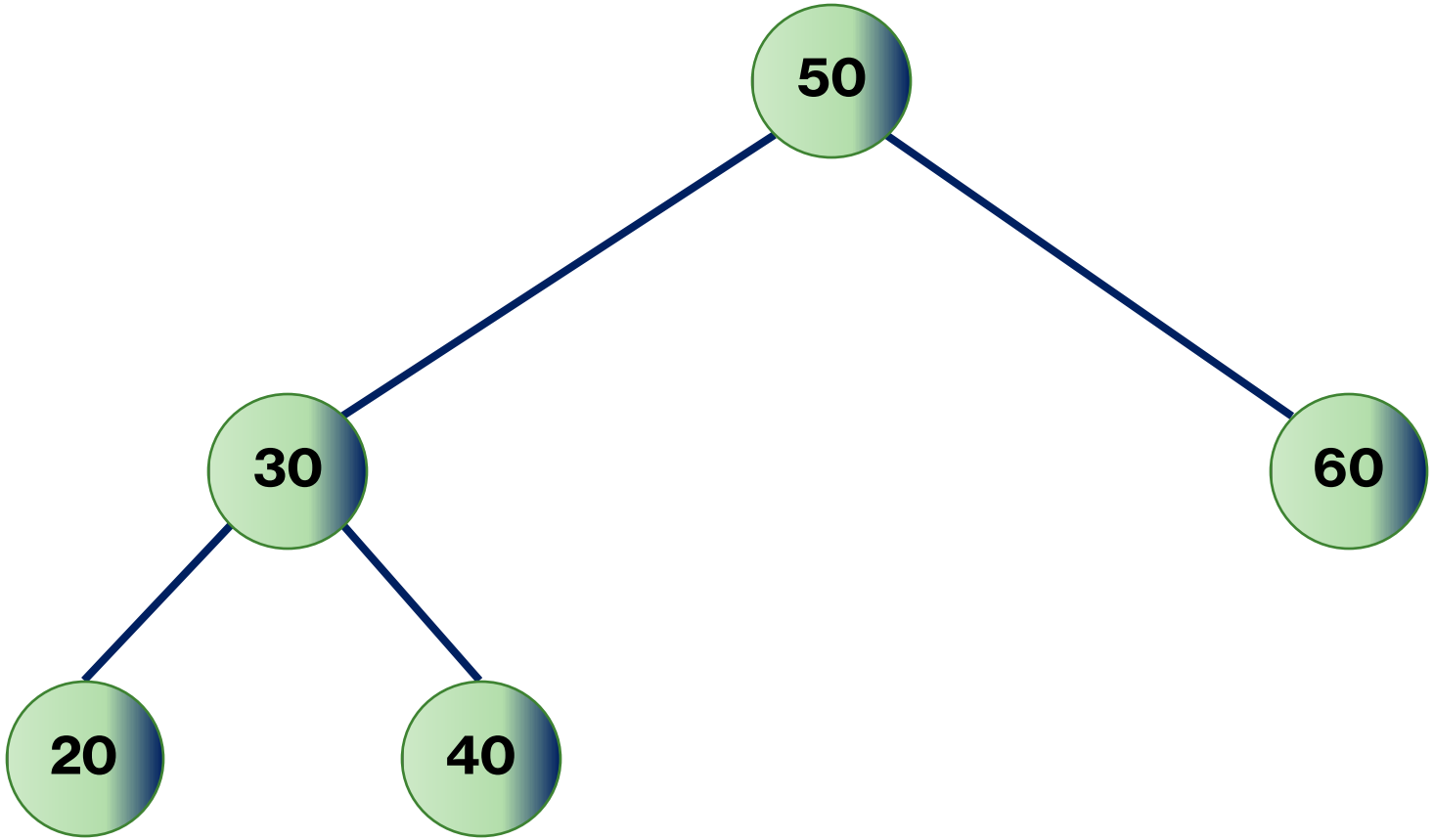


Preorder traversing

- Visit the node
- Call itself to traverse the node's left subtree
- Call itself to traverse the node's right subtree

Question 2

- Write the output, if the following tree is traverse in preorder.



Postorder traversing

01

Call itself to traverse the node's left subtree

02

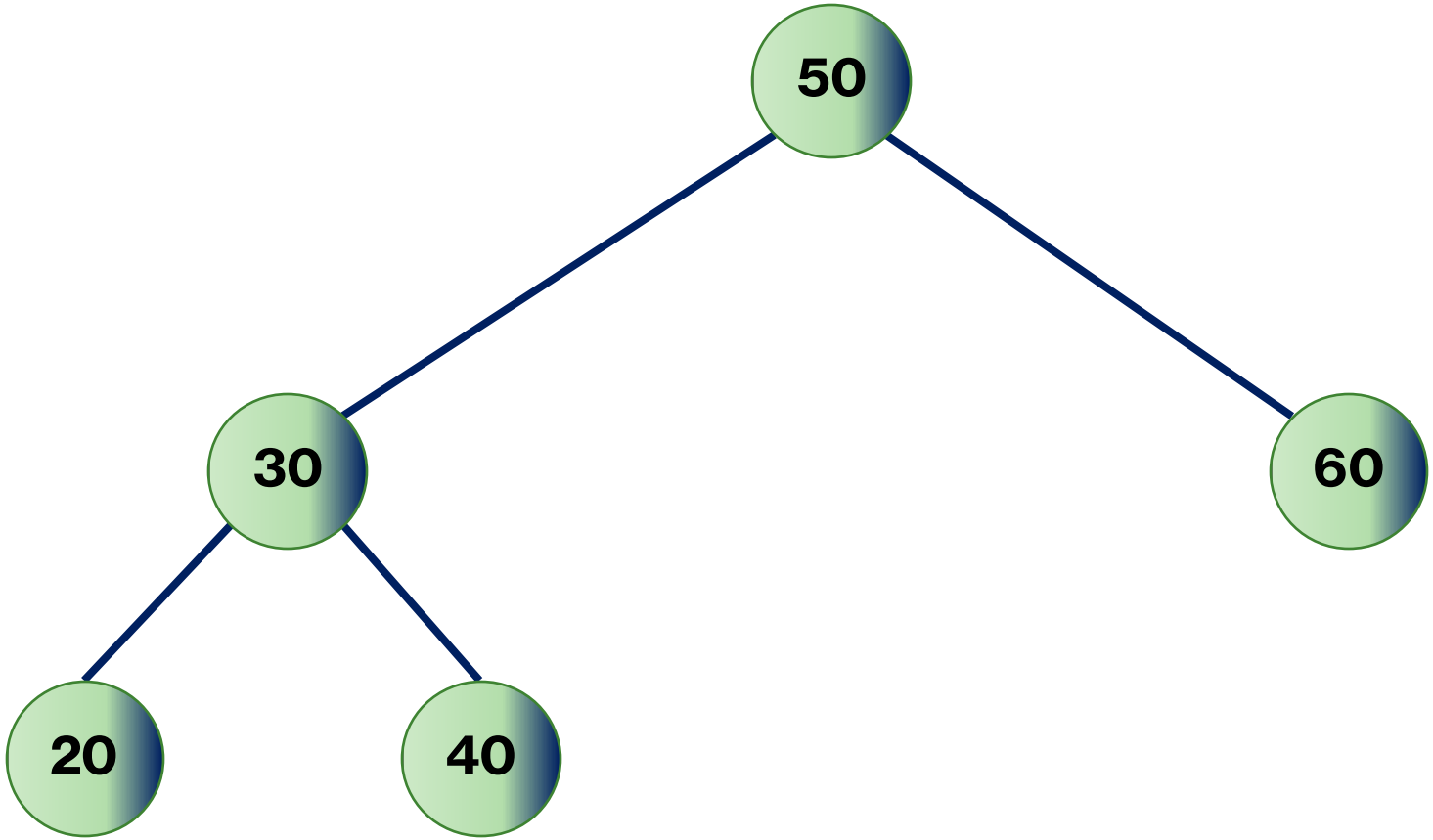
Call itself to traverse the node's right subtree

03

Visit the node

Question 3

- Write the output, if the following tree is traverse in postorder.

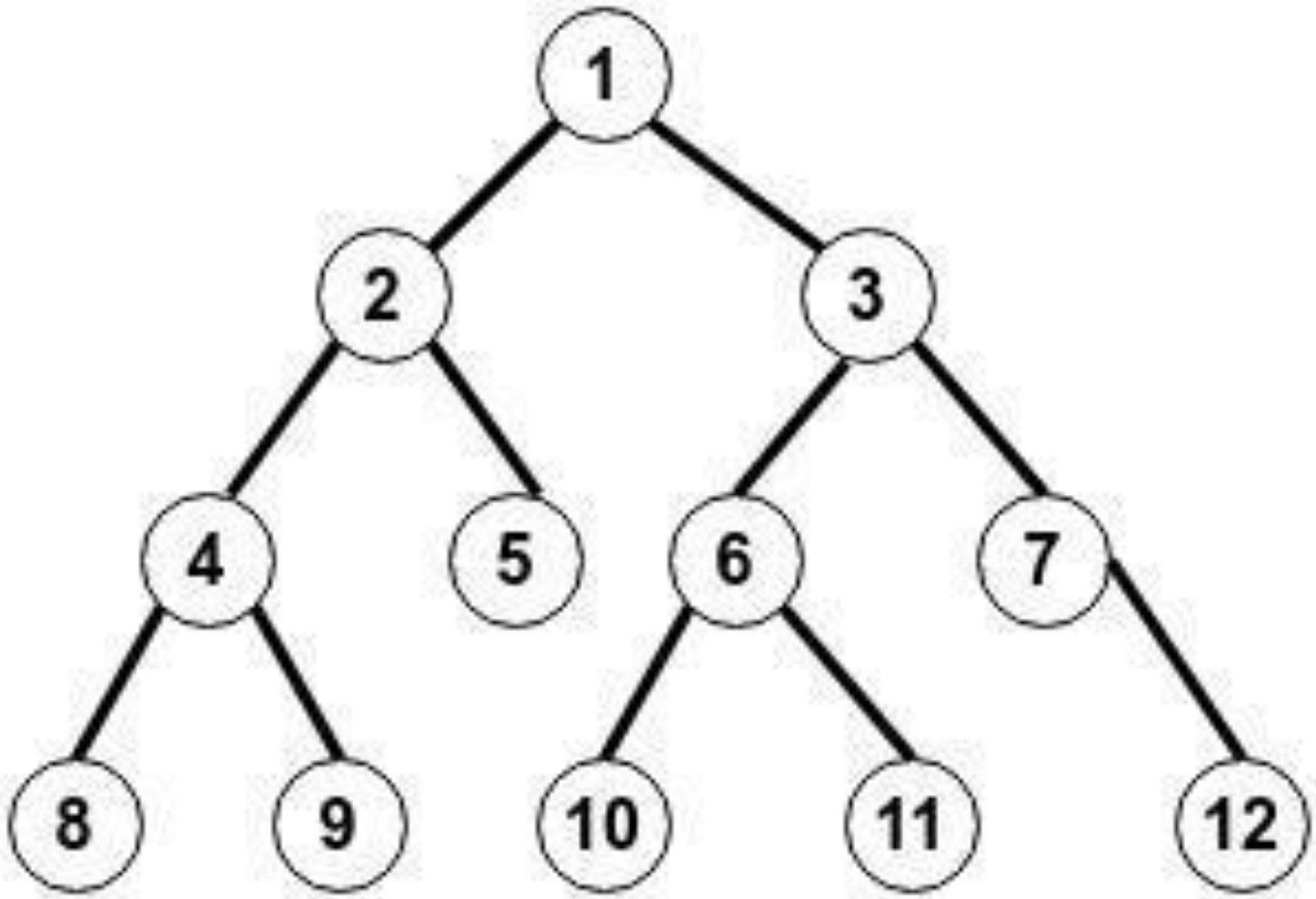


Question 4

- What is the main difference between a binary tree and a binary search tree?

Question 5

- Draw the memory representation of the following binary tree.



Question 6

Draw the binary tree having the following memory representation.

ROOT = 3

AVAIL = 15

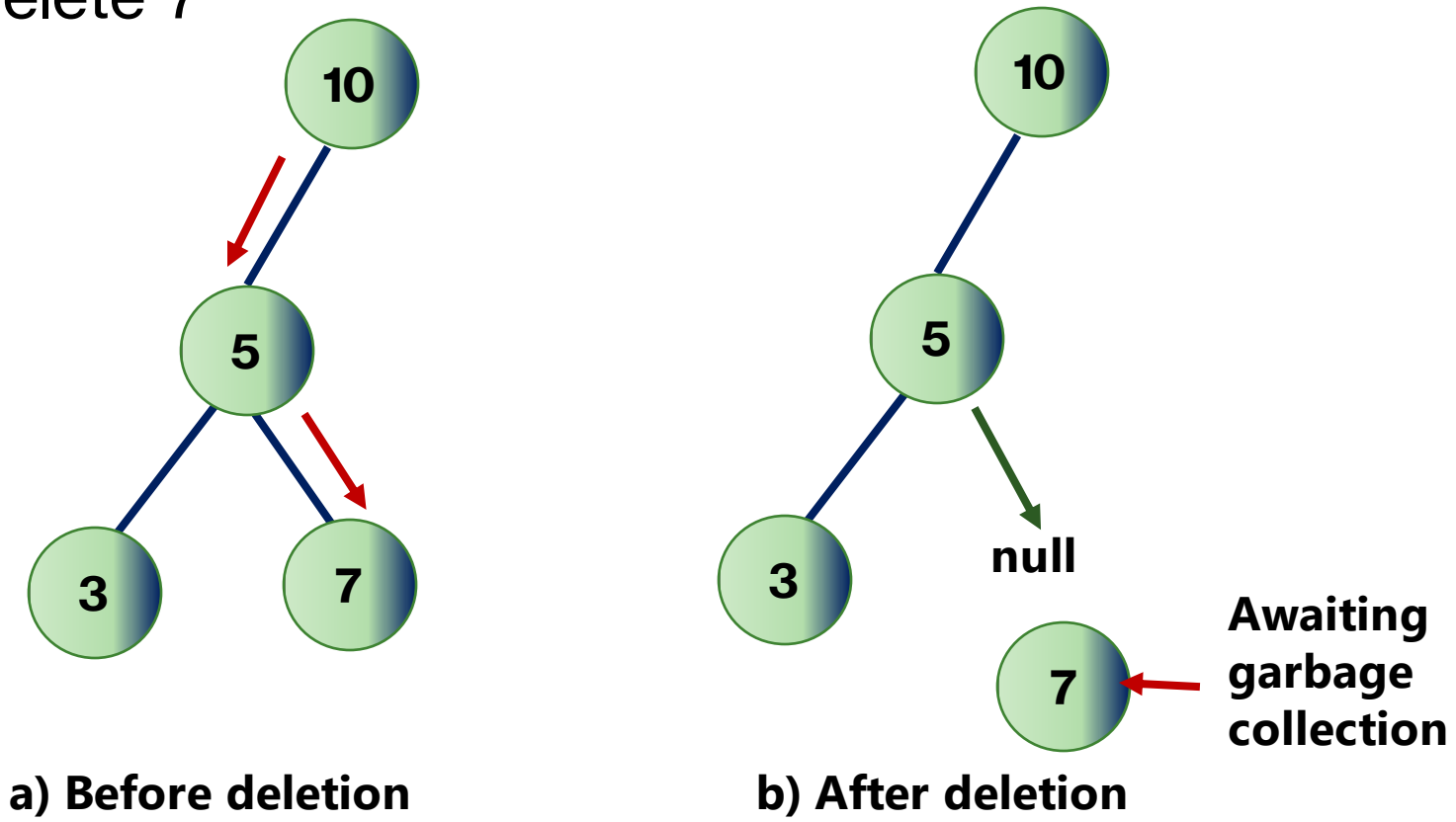
	Left	Data	Right
1	-1	8	-1
2	-1	10	-1
3	5	1	8
4			
5	9	2	14
6			
7			
8	20	3	11
9	1	4	12
10			
11	-1	7	18
12	-1	9	-1
13			
14	-1	5	-1
15			
16	-1	11	-1
17			
18	-1	12	-1
19			
20	2	6	16

Operations - DeleteOperations - Delete

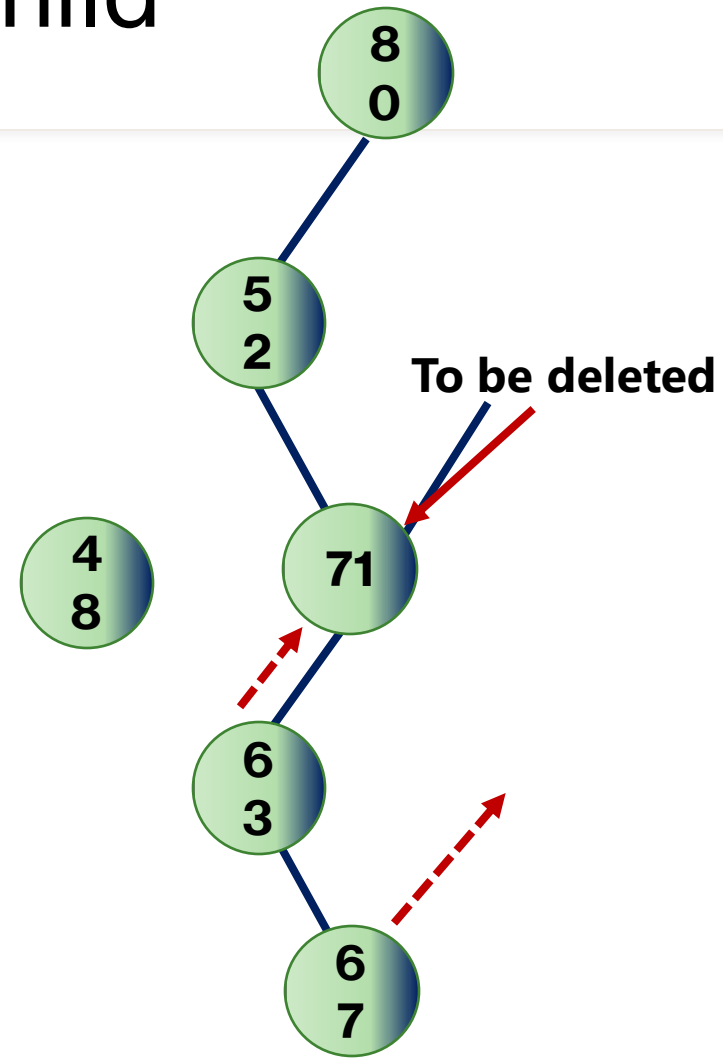
- First find the node to be deleted.
- If the node to be deleted is found there are three cases to be considered. Whether
  - The node to be deleted is a leaf
  - The node to be deleted has one child
  - The node to be deleted has two children.

Case 1 : The node to be deleted has no children

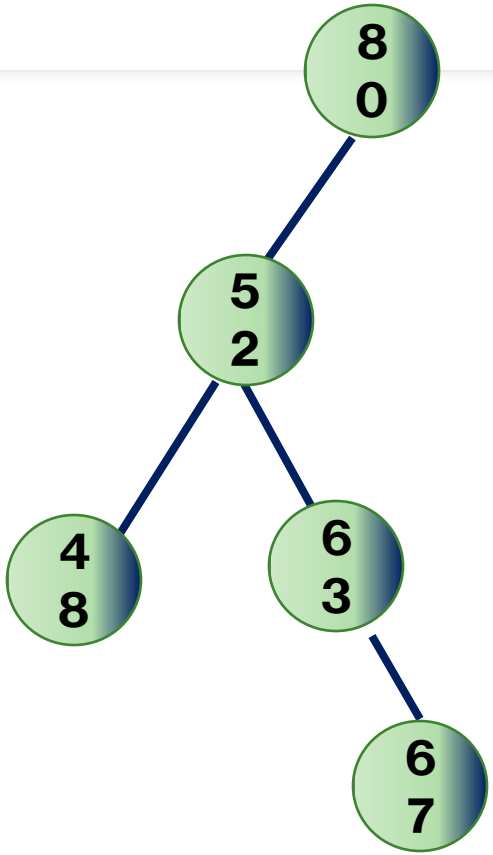
Delete 7



Case 2 : The node to be deleted has one child



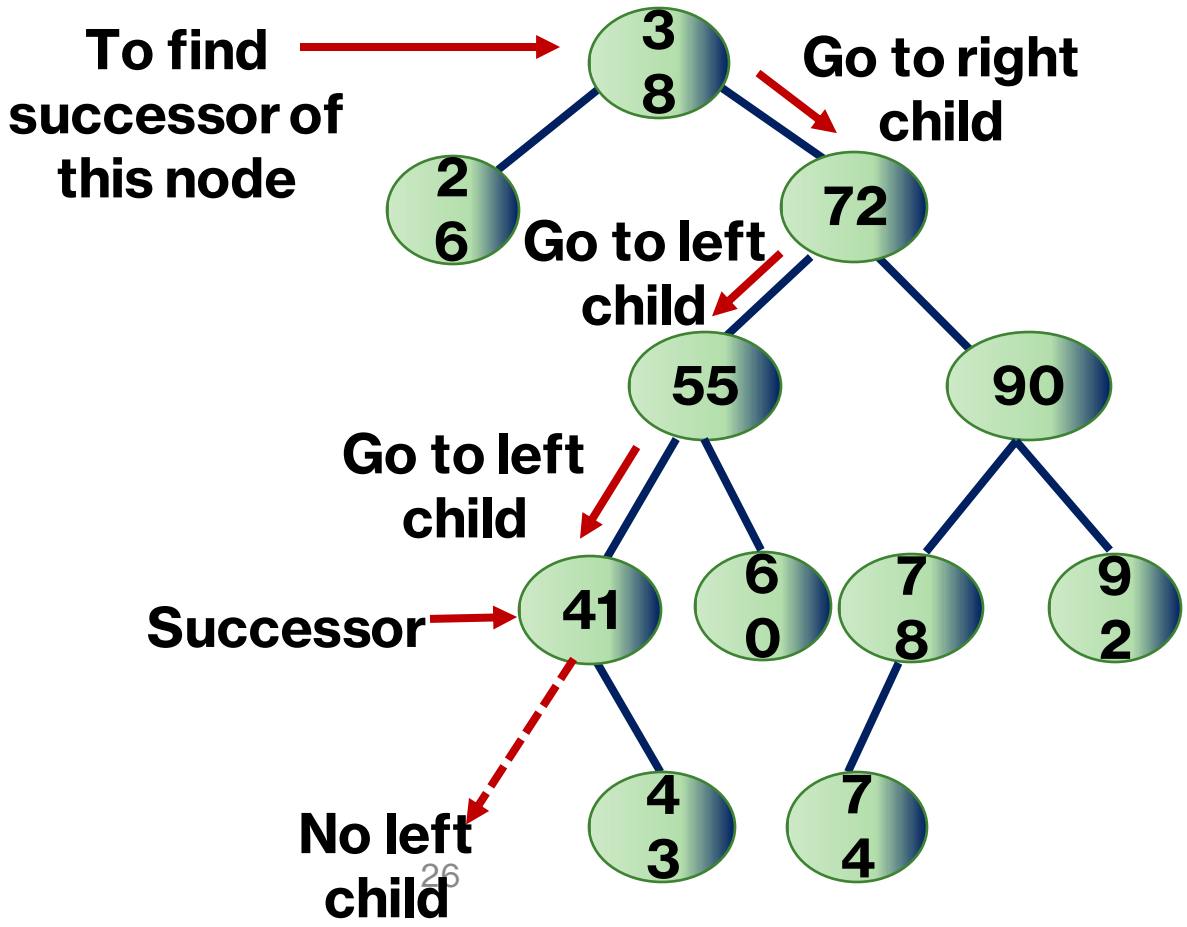
a) Before deletion



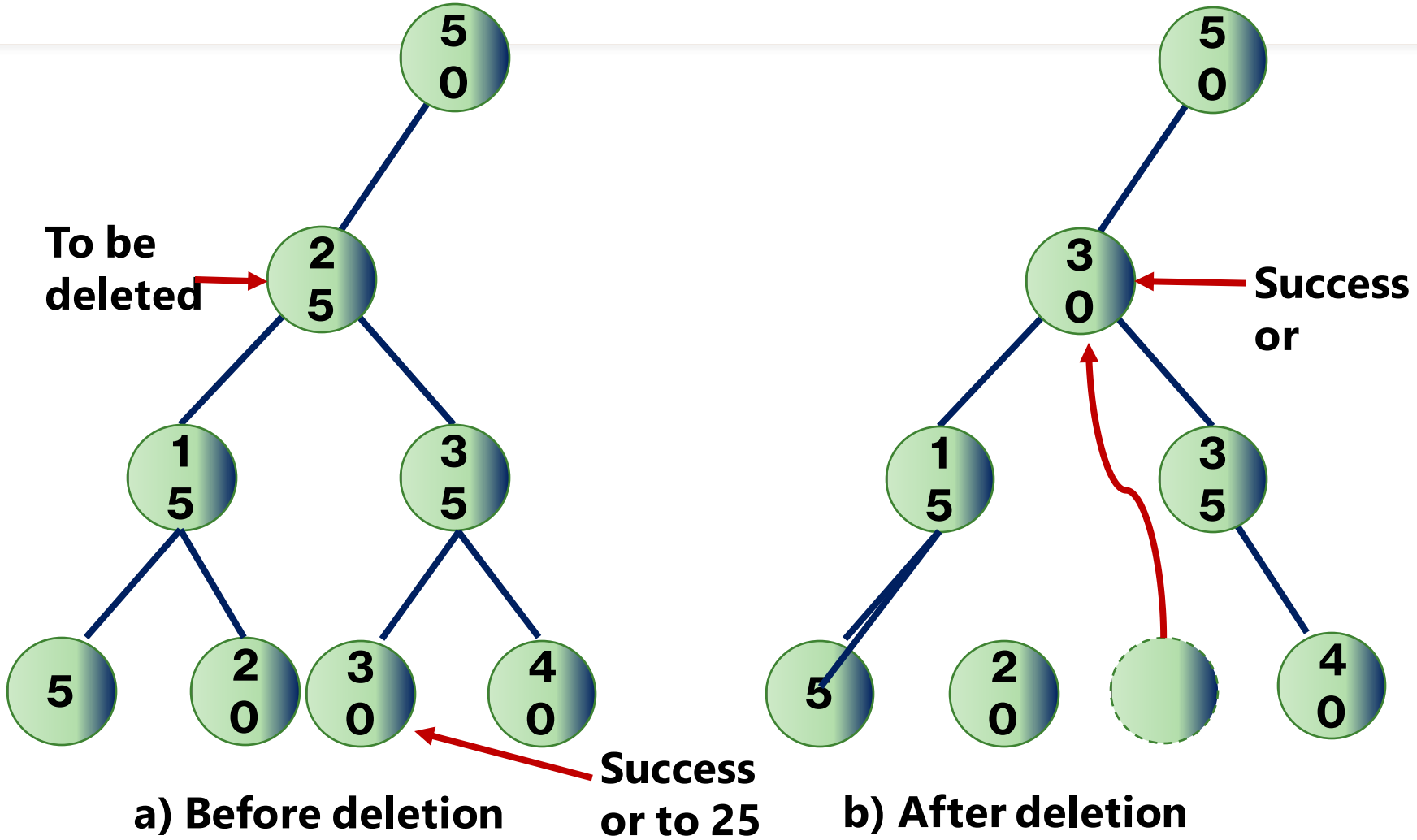
b) After deletion

How to find a successor of a node?

- In a Binary Search Tree, successor of a node is a node with next-highest key.



Case 3 : The node to be deleted has two children



Question 07

- Write a C program to build a, Binary Search Tree

- Write C programs to build,

1. Inorder Traversing Method
2. Preorder Traversing Method

Build a Binary Search Tree

Create a Node

```
struct btnode
{
    int value;
    struct btnode *l;
    struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;
```

Insert Node in the tree

```
/* To insert a node in the tree */
void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}
```

Insert data inside the Node

```
void create()
{
    int data;

    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}
```

Search the appropriate position to insert the new node

```
/* Function to search the appropriate position to insert the new node */
void search(struct btnode *t)
{
    if ((temp->value > t->value) && (t->r != NULL))      /* value more than
        root node value insert at right */
        search(t->r); //compare with the next right child node
    else if ((temp->value > t->value) && (t->r == NULL))
        t->r = temp;//if next right child node is a leaf, then make it the
            temp's right value
    else if ((temp->value < t->value) && (t->l != NULL))    /* value less
        than root node value insert at left */
        search(t->l);
    else if ((temp->value < t->value) && (t->l == NULL))
        t->l = temp;
}
```

In-Order:- It displays first left node, then root node and then right node.

```
/* recursive function to perform inorder traversal of tree */
void inorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    if (t->l != NULL)
        inorder(t->l);
    printf("%d -> ", t->value);
    if (t->r != NULL)
        inorder(t->r);
}
```

Pre-Order:- It displays in order. First root node, then left node and then right node.

```
/* To find the preorder traversal */
void preorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display");
        return;
    }
    printf("%d -> ", t->value);
    if (t->l != NULL)
        preorder(t->l);
    if (t->r != NULL)
        preorder(t->r);
}
```

Post-Order:- It displays first left node, then right node and then root node.

```
/* To find the postorder traversal */
void postorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No elements in a tree to display ");
        return;
    }
    if (t->l != NULL)
        postorder(t->l);
    if (t->r != NULL)
        postorder(t->r);
    printf("%d -> ", t->value);
}
```

Homework

Given a Binary Tree, convert it into its mirror.