Experiment 5

Ques1:- Program to search for a node with a given key in BST

i) Iterative function

```
func search (Node* root, int key)
{
    Node* curr = root, *parent = null
    while (curr != null && curr→data != key)
    {
        parent = curr
        if key < curr→data
            curr = curr→left
        else
            curr = curr→right
    }
    if parent = null
        KEY IS ~~NOT~~ FOUND, IT IS THE ROOT NOD
    if curr = null
        KEY NOT FOUND
    else if key < parent→data
        Given key is left node of parent node
    else
        Given key is right node of node with key
}
```

ii) Recursive Function

```
    func search(Node* root, int key, Node* parent)
  {
        if (root == null)
              KEY NOT FOUND
        if root → data = key
        {  if (parent == null)
                THE NODE WITH KEY IS ROOT NODE
           else if  key < parent → data
              GIVEN KEY IS LEFT NODE OF NODE WITH KEY
              else
                Given key is RIGHT node of node with key
        }
        if key < root → data
              return search(root → left, key, root)
      else
           return search(root → right, key, root)
  }
```

Ques2:- Returns the successor of a node x in a BST if it exits and NIL, if x has the largest key in the tree.

```
  func findMin(Node* root)
  {       while (root → left)
              root = root → left
          return root
  }
```

```
func successor(Node* root, Node*& succ, int key)
{
        if root = null
            succ = null
        if  root → data = key
        {   if root → right
                succ = findMin(root → right)
        }
        else if  key < root → data
        {   succ = root
            successor(root → left, succ, key)
        }
        else
            successor(root → right, succ, key)
}
```

Ques 3:- Insert new value into a BST

```
Node * insert(Node * node, int key)
{
    if (node = = NULL)
    return newNode(key)
    if key < node → key
      node → left = insert(node → left, key)
    else if  key > node → key
      node → right = insert(node → right, key)
```

```
        return node;
    }

Node * newNode(item)
{   Node * temp = memory allocation through malloc
      temp→Rey = item
      temp→left = temp→right = NULL
      return temp
}
```

Ques4:- Deleting a given node from a BST that takes as an argument a pointer to node.
Case :- i) Node to be deleted is leaf
ii) has only one child
iii) has 2 children

```
func delete(T, Z){
 if left[Z] = NULL or right[Z] = NULL
        then y ← z
         else y ← successor(z)
    if left[y] ≠ NULL
            then x ← left[y]
          else
             x ← right[y]
     if x ≠ NULL
```

```
        then  p[x] ← p[y]
    if  p[y] = NULL
        then  root[T] ← x
        else if  y = left [p[y]]
             then  left [p[y]] ← x
         else  right [p[y]] ← x
  if y ≠ z
        then key[z] ← key[y]

    // If y has other fields copy them too
  return y
}
```

Scanned with CamScanner