Page No. _____
Date

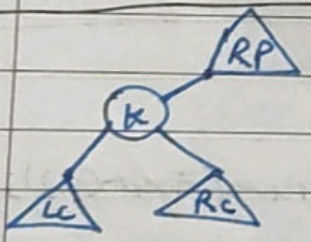
Homework 8.1

Ans⇒. Successor of any element cannot be in Left child (LC) as all elements in LC are lesser than the element itself.

- Similarly they cannot be ~~present~~ present in Left parent because elements in left parent are also less than element itself.
- If only one of Right child (RC) and Right parent (RP) are present, the successor will be contained in it.
- If both are present, then successor will be in Right child as the elements in RC will be greater than element itself but lesser than elements in RP of element.

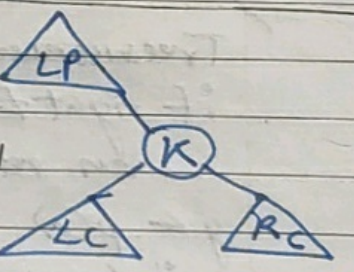
i.e. Element < elements in < elements in
itself Right child right parent

1.



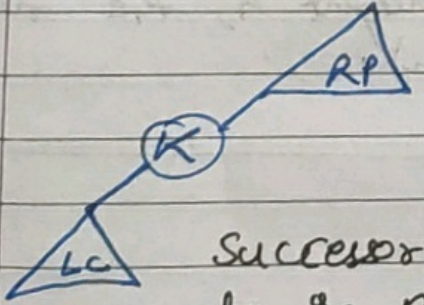
Successor will be in RC

2.



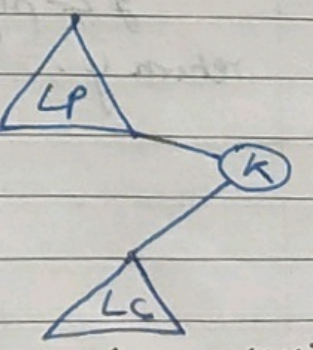
Successor will be in RC

3.



Successor will be in RP

4.



successor ~~is~~ is not present.

Homework 8.2

Ans => Observe that inOrder traversal of a BST gives output in sorted order.

We will insert elements in BST and do an inorder traversal on it.

```
Algorithm inOrder (Node n)
    if (n = NILL) then return n
    else do inOrder (n.left)
            print (n.data)
            inOrder (n.right)
```

```
Algorithm sortedOutput (int ar[])
BST searchTree ← new BST()        // Declaring BST
    for(int i=0; i<ar.length; i++)
        searchTree.insert(ar[i])  // inserting elements of BST  O(n)
    inOrder(searchTree.root)
```

In worst case, height can be O(n), therefore, above method would be O(n²) in worst case.

$$\sum_{i=1}^{n} (i-1) = \frac{n(n+1)}{2} - n = \frac{n(n-1)}{2}$$

Page No.
Date

Homework 8.3

Ans:- Queues will be the best choice.

Pseudo we will follow following method for implementation.

- Print the data corresponding to a node.
- enqueue left and right nodes (in order) in the queue.
- Perform above steps for the time when queue is non-empty.

-