DS LAB ASSIGNMENT 1 : THREADED BINARY SEARCH TREE

LOGIC EXPLAINATION OF EACH GIVEN FUNCTION --

1.Insert(val) :It will ask how many elements you want to enter after entering n elements it will make threaded binary search tree by checking the conditions and update the threads. for first element it will make it root.its time complexity is O(n).

2.Search(key) :it will compare the key starting with the root and moves left if it is less and moves right if it is more and also updates the pointer for the key ,if key gets equal to value of pointer it will return its refrerence. It is using binary search for searching in tree therefore its time complexity is log(n);

3.Delete(val) :it will first find the element and delete fron the tree and make the changes in pointers and threads accordingly.

4.ReverseInorder( ) :It will first finds the rightmost elment from the tree and then traverse using leftthreads of the tree using the left pointer. Its time complexity is O(n)

5.Succesor(ptr) :It will return the sucessor of node given ptr by checking the rightthreads and using right pointer of nodes, time complexity is log n .

6.Split(k) :saving reference of k in node by searching k in the tree using search(k);

- **Root1=root**
- **If(root>node){root2=root->left,root->left=predeccesor(node)}**
- **If(root<node){root2=root->right,root->right= Succesor (node)}**
- **If(root==node){root2=root->right,root->right=Null}**
- **If(successor(node)==rightmost(root) or predessessor(node)==leftmost(root)) {root2=successor(node) orroot2=predeccessor(node)}**
- **If(node == rightmost(root) or leftmost(root))tree is already there return root**
- **If(successor(node)->left->val <= node->val) successor(node)->left=Null**
- **Successor(successor(node))->left= successor(node)**
- **Node->right=Null;**
- **Its time complexity is O(h);**

7.AllElementsBetween (k1,k2) :it will first find elements k1 and k2 from the tree and then traverse and then traverse from k1_pointer to k2_pointer and saves the values inside a linked list.Its time complexity is O(h +n) because it finds k1 and k2 using search() which runs in O(h) and then traverse from k1_pointer to k2_pointer in O(n).

8.KthElement(k) :It uses the count of size of right sub tree saved in the node and compares the count with given parameter k and if value of count of that node equals to k the it will returns the value of node.Its time complexity is O(h).

9.PrintTree( ) : It will draw the graph by using GRAPHVIZ by reading the file graph.gv .

TEST CASES AND OUTPUTS—

- **Insert(val)**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 1
How many elements you want to insert:13
10
80
4
67
3
41
7
90
1
21
95
63
100
```

- **Search(key)**

```
Do you want to continue (Type y or n): y

Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 3
Enter integer element to search: 7
Element 7 found in the tree and its REFERENCE IS :0x6e7b00

Do you want to continue (Type y or n):
```

- **Delete(val)**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 2
Enter integer element to delete: 10

Do you want to continue (Type y or n): y

Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 9

PRINTING TREE USING GRAPHVIZ
1 3 4 7 21 41 63 67 80 90 95 100
```

- **ReverseInorder( )**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 4

PRINTING REVERSE INORDER
100 95 90 80 67 63 41 21 10 7 4 3 1
```

- **Succesor(ptr)**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 5

PRINTING INORDER SUCCESSOR
ENTER THE ELEMENT :90

INORDER SUCCESSOR IS=
95
Do you want to continue (Type y or n):
```

- **Split(k)**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 1
How many elements you want to insert:13
10 80 4 67 3 41 7 90 1 21 95 63 100

Do you want to continue (Type y or n): y

Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 6

SPLITING THE TREE
ENTER K :4

 split


root1=10
root2=4
```

- **AllElementsBetween (k1,k2) :**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 7

PRINTING ALL ELEMENTS BETWEEN K1 AND K2 (K1 ,K2 INCLUDED)
ENTER K1 4

ENTER K2 41
4 7 10 21 41
```

- **KthElement(k)**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 8

PRINTING Kth LARGEST ELEMENT

ENTER K 5

Kth LARGEST ELEMENT IS :67
Do you want to continue (Type y or n):
```

- **PrintTree( )**

```
Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 1
How many elements you want to insert:13
10 80 4 67 3 41 7 90 1 21 95 63 100

Do you want to continue (Type y or n): y

Threaded_BinarySearchTree_Operations
1. Insert
2. Delete
3. Search
4. Reverse Inoder
5. Successor
6. Split
7. All Elements Between
8. Kth Largest Element
9. Print tree
Enter Your Choice: 9

PRINTING TREE USING GRAPHVIZ
1 3 4 7 10 21 41 63 67 80 90 95 100
Do you want to continue (Type y or n):
```