

## BSTnode structure-

Ltag	*left	Data	*right	Rtag	rcount
------	-------	------	--------	------	--------

Ltag=-1 left pointer points to the NULL.

Ltag=1 left pointer points to the predecessor.

Ltag=0 left pointer points to subtree

Rtag=-1 Rightpointer points to the NULL.

Rtag=1 Right pointer points to the predecessor.

Rtag=0 Right pointer points to subtree.

Rcount : count the number of node present in right subtree.

### 1. insert (x):-

#### **Case-1**

When element inserted in tree first it checks is tree empty or not if tree is empty then a node is inserted in tree and Ltag and Rtag is -1 because no successor and predecessor is present in tree.

#### **Case-2**

##### **Inserted as left node-**

Ltag of child is same as Ltag of parent after that we modify the Ltag of parent node (now Ltag of parent is 0) because parent now point to the child node. Left pointer of the node points to the same element which his parent points previously because the predecessor of the child node is the same node which the parent left pointer points previously. Now the parent node left pointer points to the new child node.

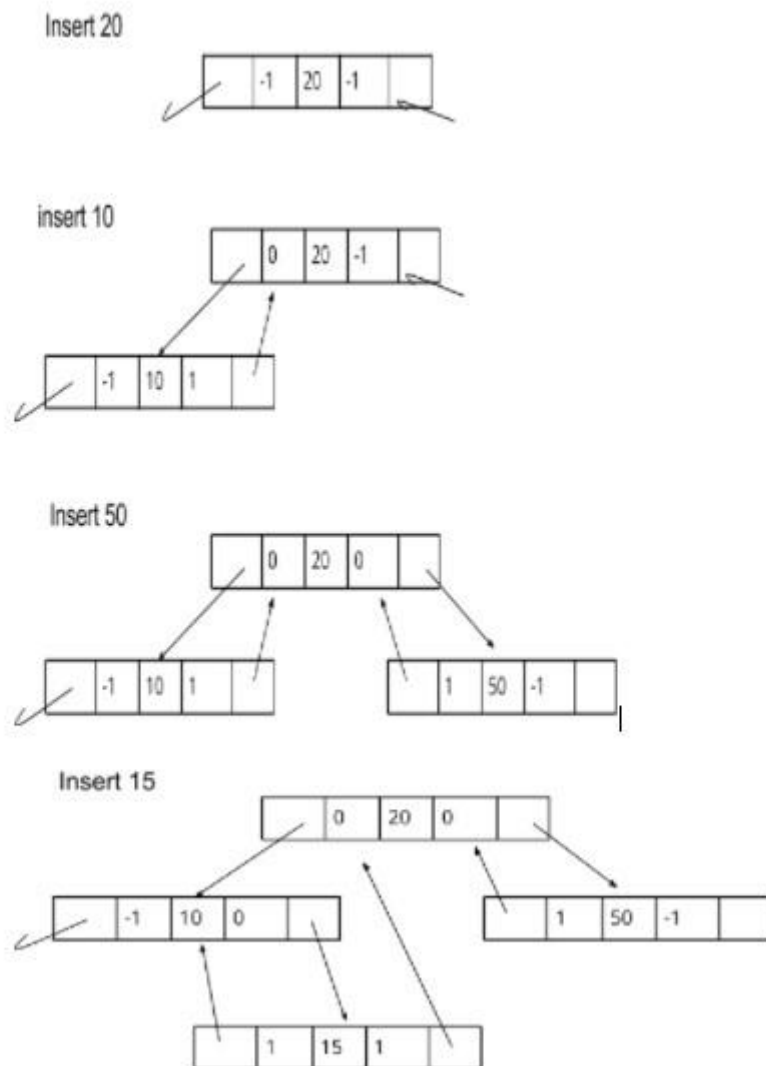
Rtag of child is 1 because it points to the successor and successor of the newly inserted element is his parent.

##### **Inserted as right node -**

Rtag of child is the same as Rtag of parent. After that we modify the Rtag of parent node (now Rtag of parent is 0) because parents now point to the child node. Right pointer of the node points to the same element which

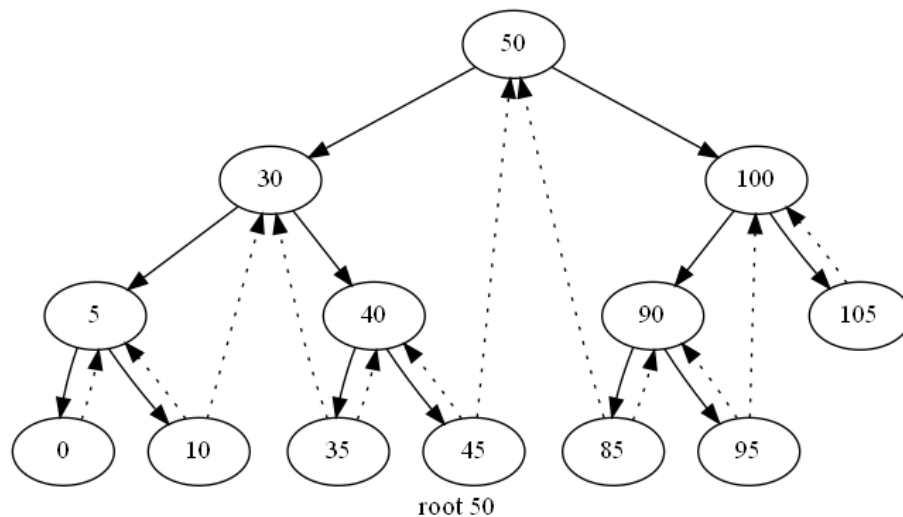
his parent points previously. Now the parent node right pointer points to the new child node.

Ltag of child is 1 because it points to the predecessor and predecessor of the newly inserted element is his parent.



Insert element in tree -

50,30,100,5,40,90,105,0,10,35,45,85,95



## 2. Search(x)-

Whenever we want to search an element we start from the root node. If  $x$  (means search element) is less than the key value of root then we search in the left subtree and if the search element is greater than the key value of root we search in the right subtree. If we find the end point of the tree means when the tag is not equal to 0 then we print "Element is not present in Tree".

```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018
Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. sucesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 2
0 5 10 30 35 40 45 50 85 90 95 100 105
Enter Search Element :40
Element present :
Enter Your choice - 3
0 5 10 30 35 40 45 50 85 90 95 100 105
Enter element :15
Element is not present in tree :0 5 10 30 35 40 45 50 85 90 95 100 105
Enter Your choice -
```

### 3. delete(x)-

steps-

Find element is present in tree or not

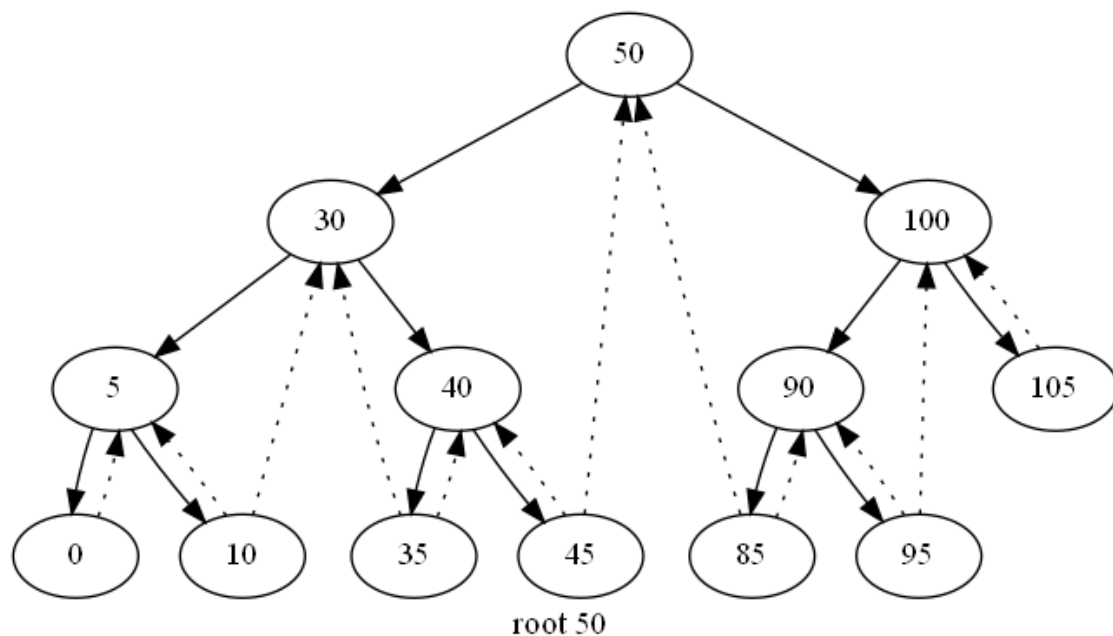
If element is present, then find the parent and current node (the node which we want to delete) .

If current node have two child then the find the successor or node and replace with the successor element. We modify the parent of successor and also successor of parent node.

If current node have one child we delete that node and modify the successor predecessor of nodes

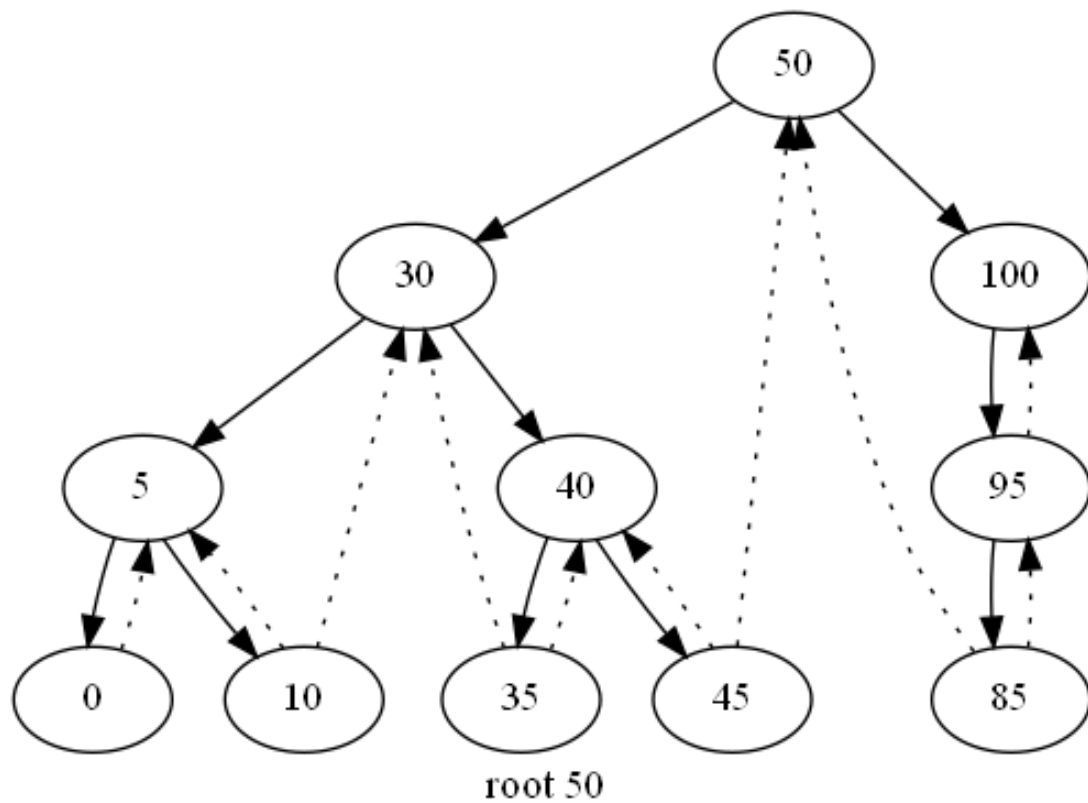
```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018

Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. sucesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 3
0 5 10 30 35 40 45 50 85 90 95 100 105
Enter element :30
0 5 10 35 40 45 50 85 90 95 100 105
Enter Your choice - 3
0 5 10 35 40 45 50 85 90 95 100 105
Enter element :12
Element is not present in tree :0 5 10 35 40 45 50 85 90 95 100 105
Enter Your choice -
```



Delete(90)

Delete(105)



#### 4. reversalInorder()-

For finding the reversal inorder-

- we first find the maximum element that means right most node of the tree.
- After that we start inserting elements in the link list and find the in-order predecessor of that node next node is inorder predecessor
- insert a next element at the last position of the linked list until all elements of the tree are inserted.

```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018
Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. succesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 4
0 5 10 30 35 40 45 50 85 90 95 100 105
Reversal inorder is :
105 100 95 90 85 50 45 40 35 30 10 5 0
Enter Your choice -
```

## 5. successor(ptr)-

### Case 1-

1. If element have not right child then Rtag of pointer is 1 means it point to the successor of node so we can directly return the ptr->right;
2. If Rtag of pointer is -1 means,it's not point to anything means not successor present.
3. If Rtag=0 means element have right child, then we return the smallest element of right subtree.

## 6. split(k)-

Split the tree in two part 1<sup>st</sup> tree contain less then equal to k and other contain greater then k.

Pseudocode –

Best case-

If one node is present and  $k < \text{root key}$  then return

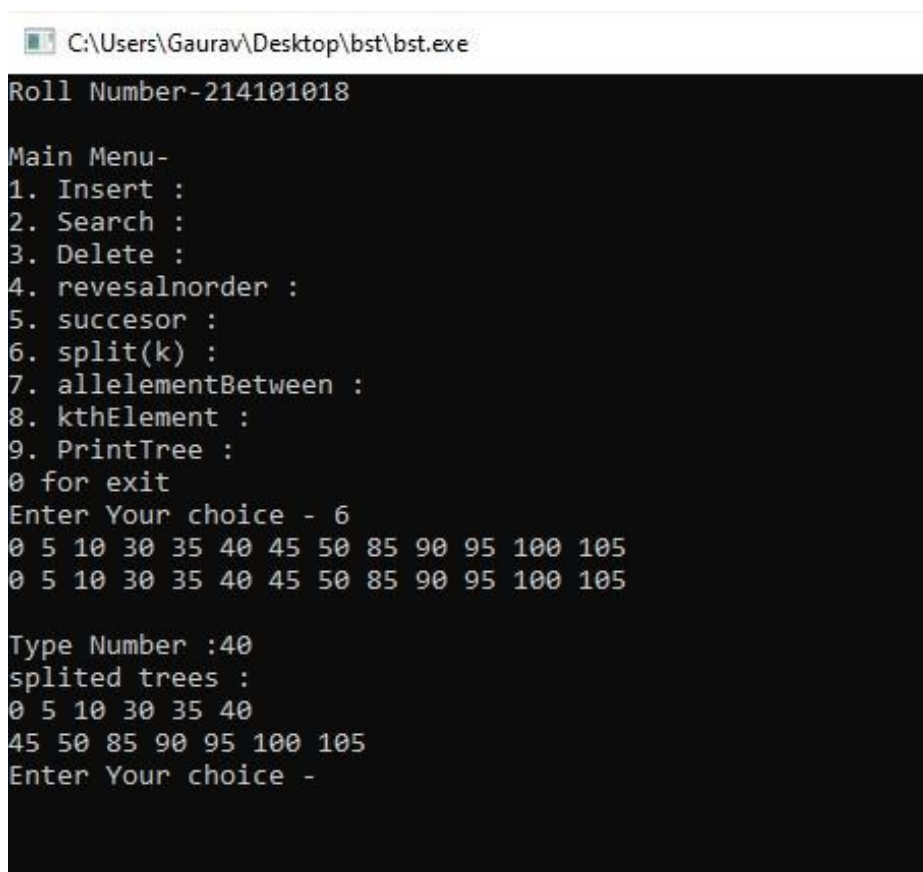
$L = \langle \text{NULL}, \text{key} \rangle$

If one node is present and  $k \geq \text{root key}$  then return

$L = \langle \text{key}, \text{NULL} \rangle$

If both node is present and let  $k < \text{root key}$  then split call on right subtree and  $\text{root} \rightarrow \text{right} = L[0]$  and return  $\langle \text{root}, L[1] \rangle$

If both node is present and let  $k > \text{root key}$  then split call on right subtree and  $\text{root} \rightarrow \text{left} = L[1]$  and return  $\langle L[0], \text{root} \rangle$



```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018
Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. succesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 6
0 5 10 30 35 40 45 50 85 90 95 100 105
0 5 10 30 35 40 45 50 85 90 95 100 105

Type Number :40
splited trees :
0 5 10 30 35 40
45 50 85 90 95 100 105
Enter Your choice -
```

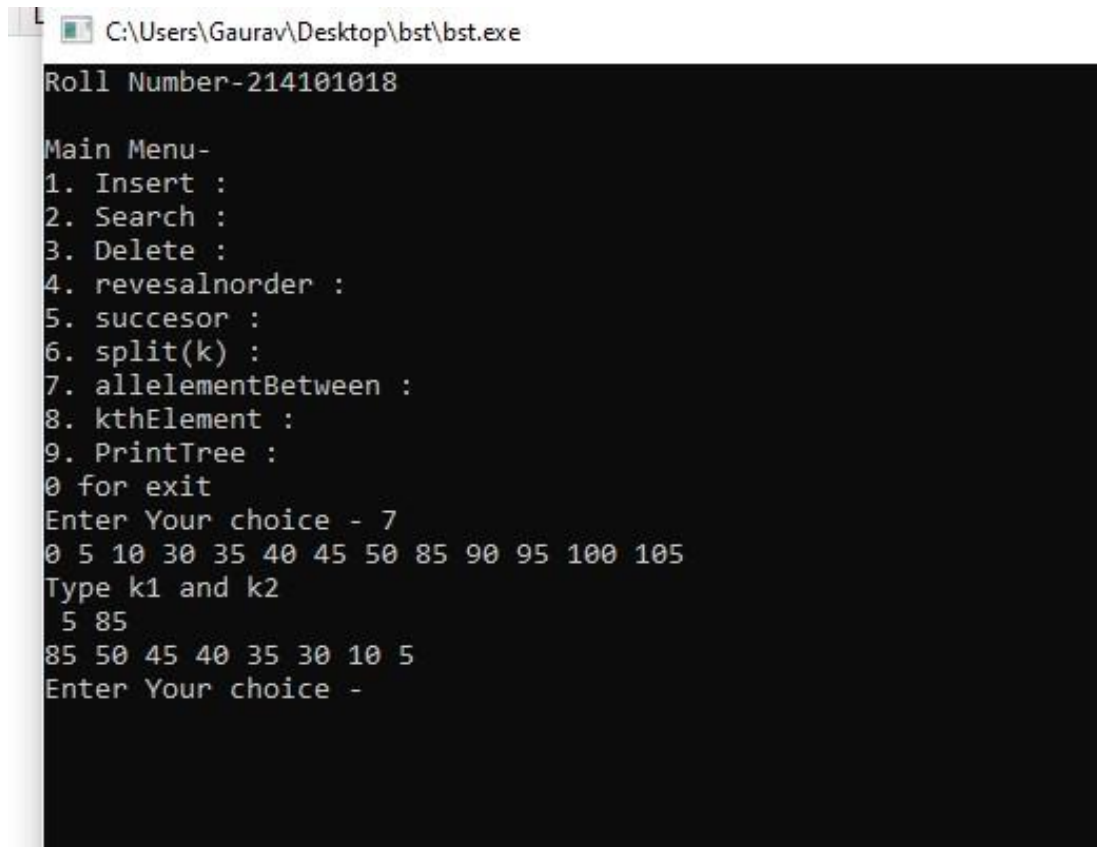
## 7. allElementsBetween(k1, k2)-

For finding elements between  $k1$  to  $k2$  we need to first find the element which is present in the tree. For this we traverse the graph from the root node if the root



key value is less than k1 then we find in the right subtree and if root key value is greater than k2 then we find in the left subtree. When we find the smallest element which is greater than k1 we start inserting an element in the linked list and every time we find the successor of the current node until the key value is less than k2.

Here k1=5 and k2=85



```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018

Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. succesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 7
0 5 10 30 35 40 45 50 85 90 95 100 105
Type k1 and k2
5 85
85 50 45 40 35 30 10 5
Enter Your choice -
```

## 8. kthElement(k) –

For finding the kth largest element of tree we store the rcount (number of node in right subtree) in every node

**Case-1 :**

If  $k = rcount + 1$  means the right subtree of that contain k-1 largest element so root node is kth largest element

**Case-2:**

If  $k < rcount$  means left subtree contain the largest element so we can directly go to the right subtree and skip the left subtree.

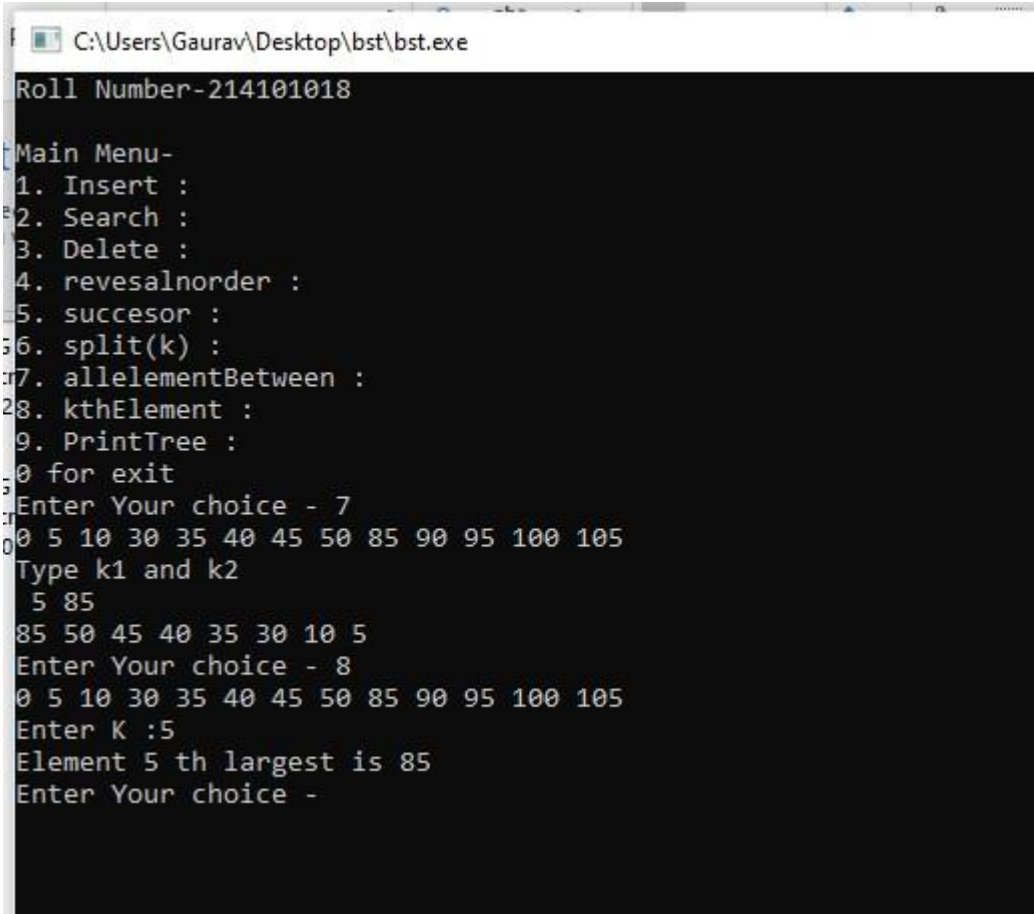
### Case-3:

If  $k > rcount+1$  means left subtree contain the kth largest element. We skip the right tree traversal with modifying k value

$$K = k - rcount + 1$$

And we find the new kth largest in left subtree

By this way we can search kth largest element in  **$O(h)$  time.**



```
C:\Users\Gaurav\Desktop\bst\bst.exe
Roll Number-214101018
Main Menu-
1. Insert :
2. Search :
3. Delete :
4. revesalnorder :
5. succesor :
6. split(k) :
7. allelementBetween :
8. kthElement :
9. PrintTree :
0 for exit
Enter Your choice - 7
0 5 10 30 35 40 45 50 85 90 95 100 105
Type k1 and k2
5 85
85 50 45 40 35 30 10 5
Enter Your choice - 8
0 5 10 30 35 40 45 50 85 90 95 100 105
Enter K :5
Element 5 th largest is 85
Enter Your choice -
```

## 9. Print():-

Store the link between two node in tree.dot file and after that we convert dot file in png

This is the tree when I insert element

50,30,100,5,40,90,105,0,10,35,45,85,95

