DS LAB Assignment 1

Roll number : 214101022

Code is written in c++.

Graphviz should be installed in pc for print tree part.

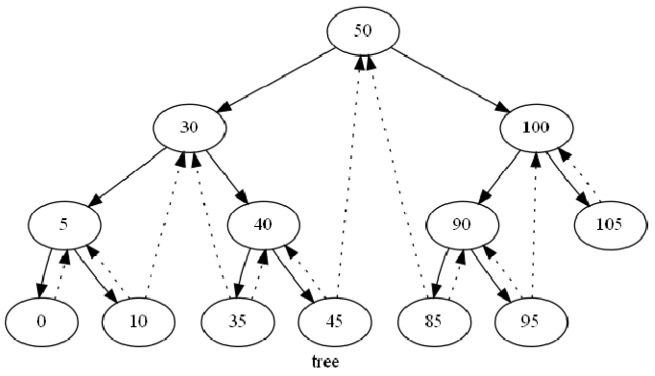
System call is added to print part so that you can see image of tree in same folder.(Check image.png)

A tree is already inserted. You can print and see it

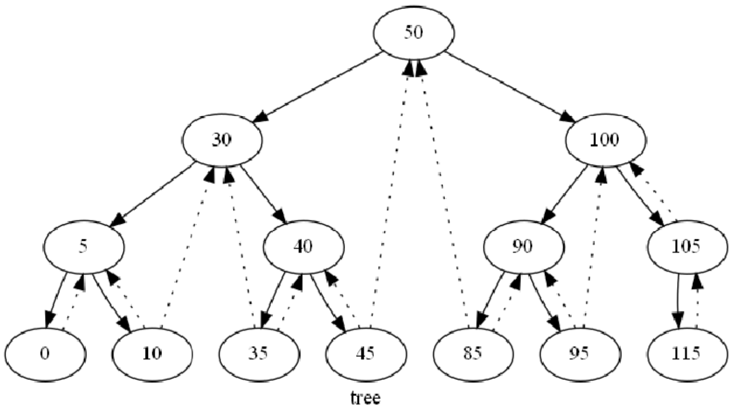
Operations:

(1) Insertion into the tree:

- Provide input to the tree.
- Function will create a new node and traverse to its proper place.
- Threads will be rearranged.
- New node will be added to the tree.
- Print the tree and check where the node is added.

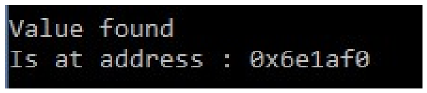


115 is added in new tree



(2) Search function:

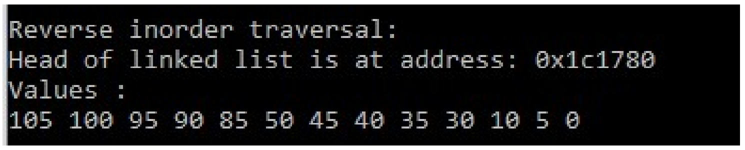
- Provide an input to the function.
- It will return pointer to the node if node is present in tree.
- Tree will traverse from root to node to find the node.
- If found, will print the address of the node.



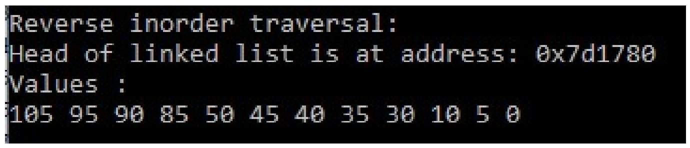
```
Value found
Is at address : 0x6e1af0
```

(3) Deletion:

- There are three cases for deletion part.
- First case : Node is leaf node. We can rearrange threads and delete the current node. Just like insertion part but in reverse order.
- Second case : Node have only one children. We can find successor and predecessor of current node. Join predecessor to its successor and delete the current node.
- Third case : Node has two children. We can find the maximum key from left sub tree or minimum key of right sub tree. This node will be either leaf node or a node with one child. Exchange the nodes and perform first/second case
- 100 is deleted in below example



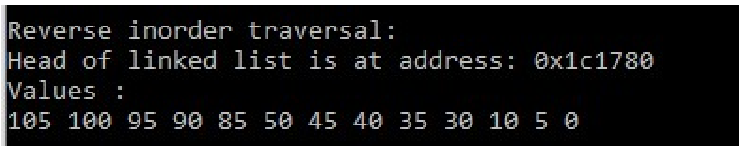
```
Reverse inorder traversal:
Head of linked list is at address: 0x1c1780
Values :
105 100 95 90 85 50 45 40 35 30 10 5 0
```



```
Reverse inorder traversal:
Head of linked list is at address: 0x7d1780
Values :
105 95 90 85 50 45 40 35 30 10 5 0
```

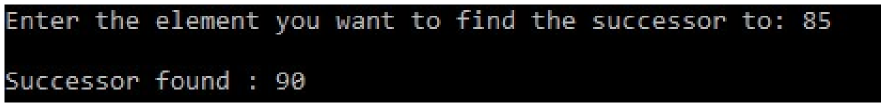
(4) Reverse In-order:

- Traverse to the rightmost node in tree. Add node in linked list.
- Use predecessor function to find its predecessor and its key to the linked list.
- Do this till we get the leftmost node in tree.
- Return the linked list head.
- Print the values.


```
Reverse inorder traversal:
Head of linked list is at address: 0x1c1780
Values :
105 100 95 90 85 50 45 40 35 30 10 5 0
```

(5) Successor:

- Traverse to the node using Search operation.
- Find minimum node in the right sub tree.
- Return the pointer of the node.

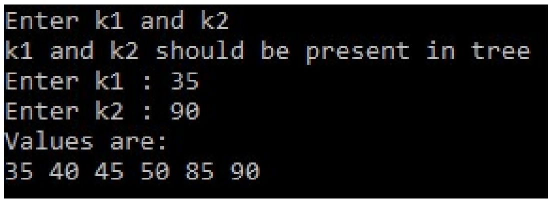

```
Enter the element you want to find the successor to: 85

Successor found : 90
```

(6) Split:

- Use copy constructor to copy current tree in new tree.
- Find the edge where the keys are less than k and greater than or equal to k. Save it somewhere. Cut the edge.
- Do it using recursion and traverse to the leaf.
- Now join all the broken tree parts into two trees where values of one tree contains keys <= k and other tree keys > k.
- Recursion will do the job here. Don't need to join trees separately.

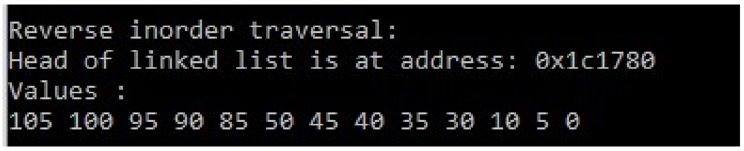
(7) All keys between k1 and k2:

- Use Search function to traverse to k1 and add key in linked list.
- Use successor function in while loop to traverse to k2.
- Return the linked list and print list.
- Can be done in O(h+N) time.

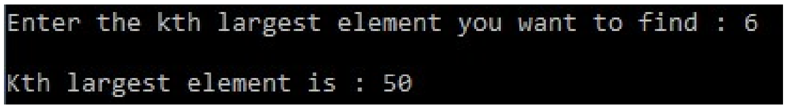

```
Enter k1 and k2
k1 and k2 should be present in tree
Enter k1 : 35
Enter k2 : 90
Values are:
35 40 45 50 85 90
```

(8) kth largest key:

- While doing insertion part keep track of number of nodes in right sub tree.
- Every node now contains the number of successors present in tree.
- Now use Search function to traverse from root to the node containing k-1 successors.
- Return the node. Can be done in O(h) time.



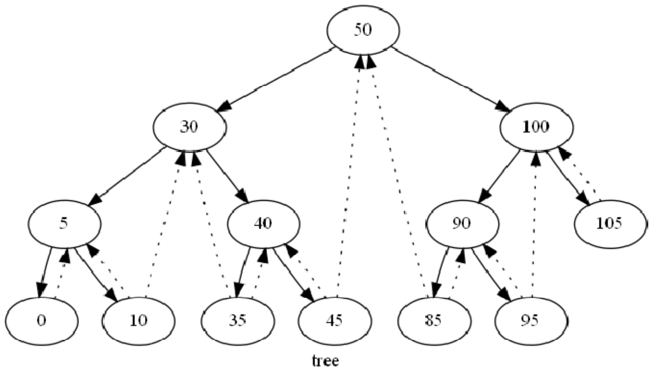
```
Reverse inorder traversal:
Head of linked list is at address: 0x1c1780
Values :
105 100 95 90 85 50 45 40 35 30 10 5 0
```



```
Enter the kth largest element you want to find : 6

Kth largest element is : 50
```

(9) Print tree:

- Install graphviz.
- Use file pointer to write all edges of the tree into the image.dot file using level order traversal.
- System is already there in code. No need to do it separately.
- Check the folder you will find image.png file.
- Image is given below of current tree.


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