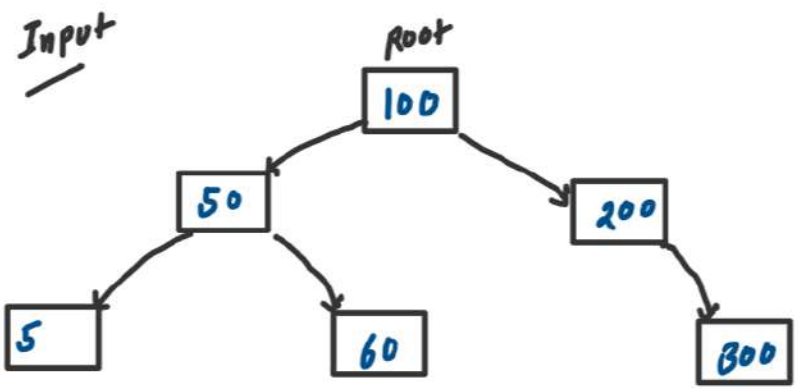
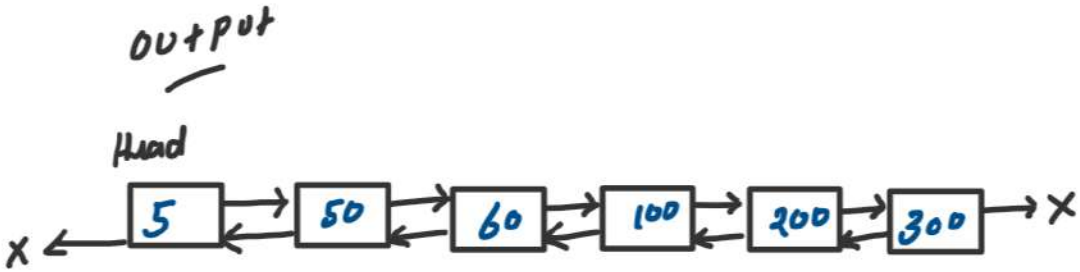
13/12/2023

BINARY SEARCH TREE
CLASS - 3

1. Convert BST into Sorted Double Linked List



BST

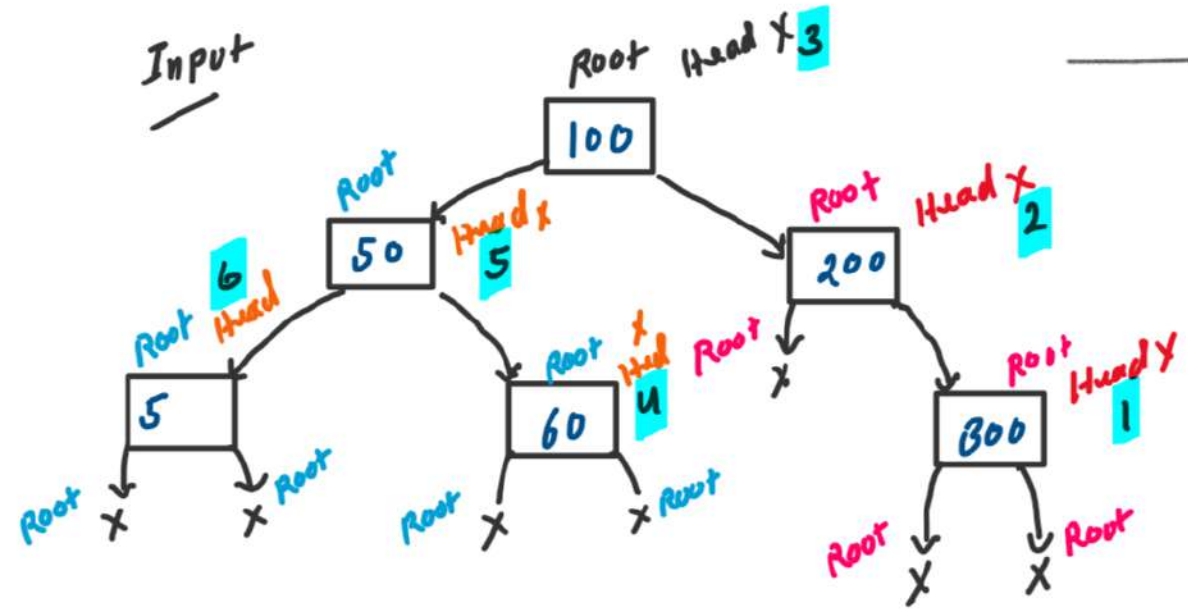


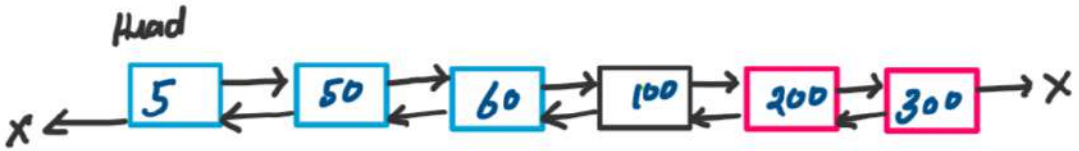
DLL

Logic Building

SOLVE USING
REVERSE INORDER TRAVERSAL
(RNL)

↳ initially Head = NULL

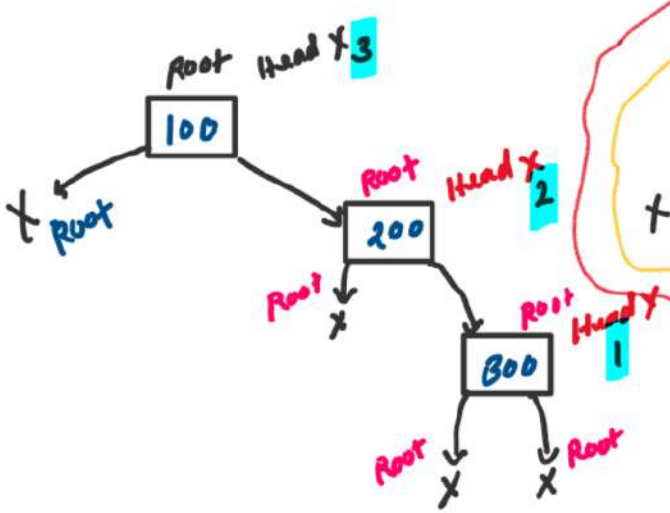


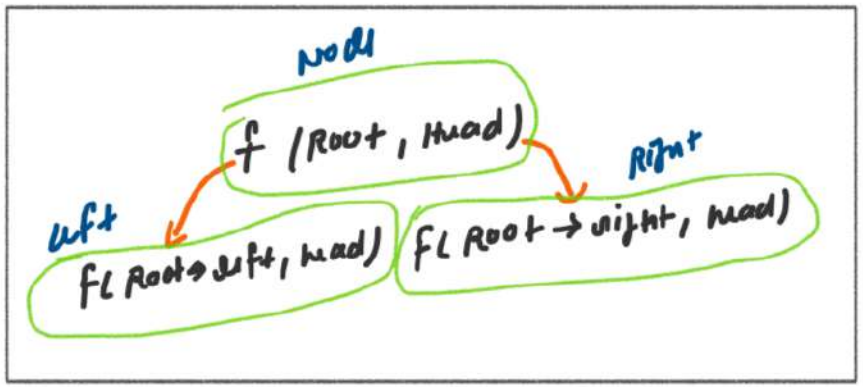


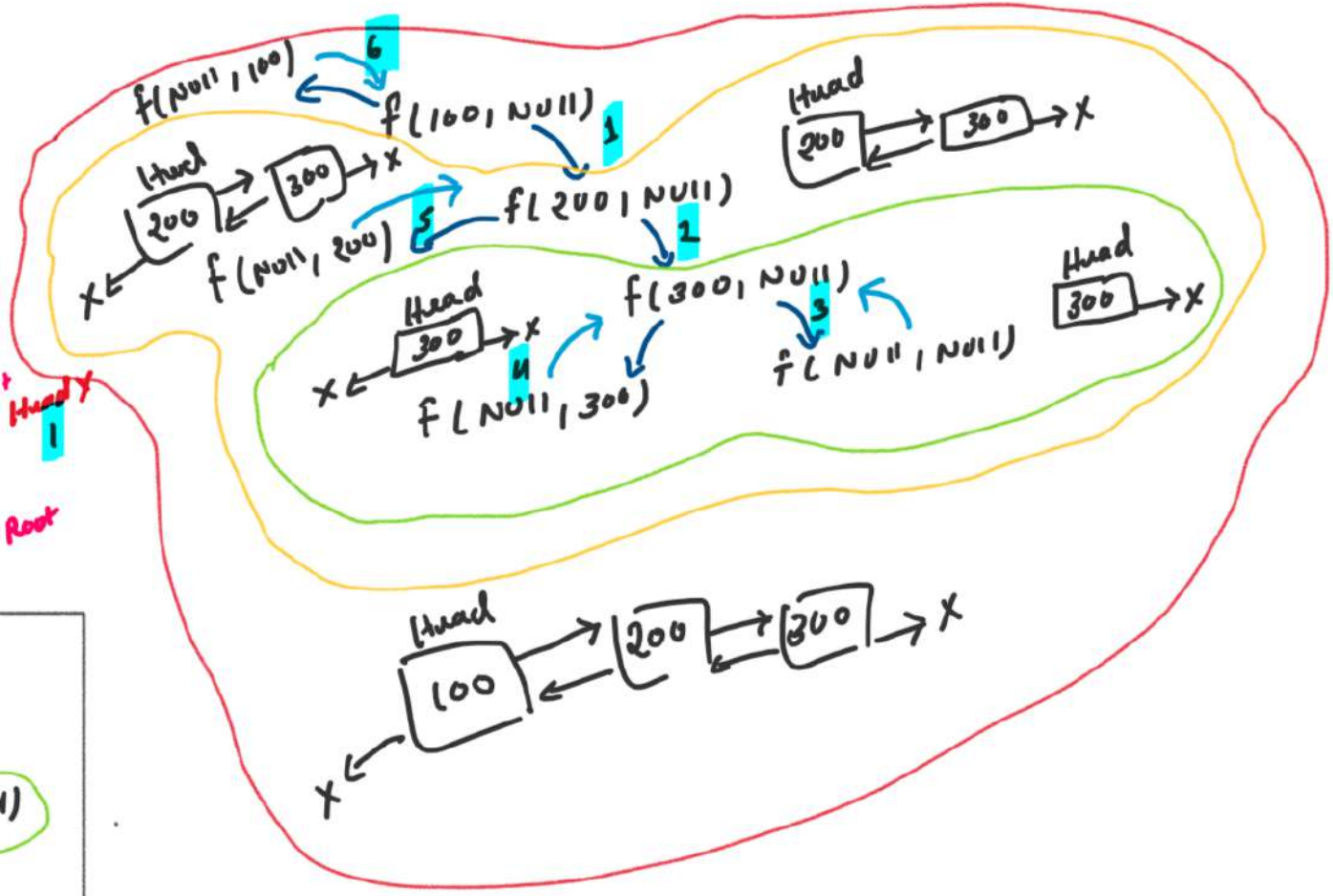

```
Right   fl(root->right, head)

NODE    root->right = head;
        if(head != NULL){
            head->left = root;
        }
        head = root;

Left    fl(root->left, head)
```

DRY RUN

EXL

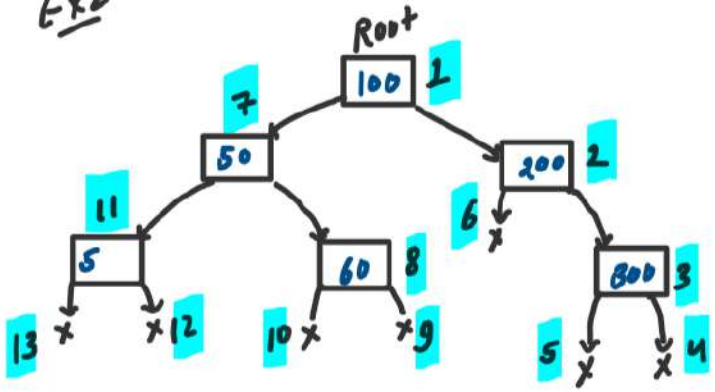

Root
Head
3
100
Root
X
Root
Head
2
200
Root
X
Root
Head
1
300
Root
X
Root
X

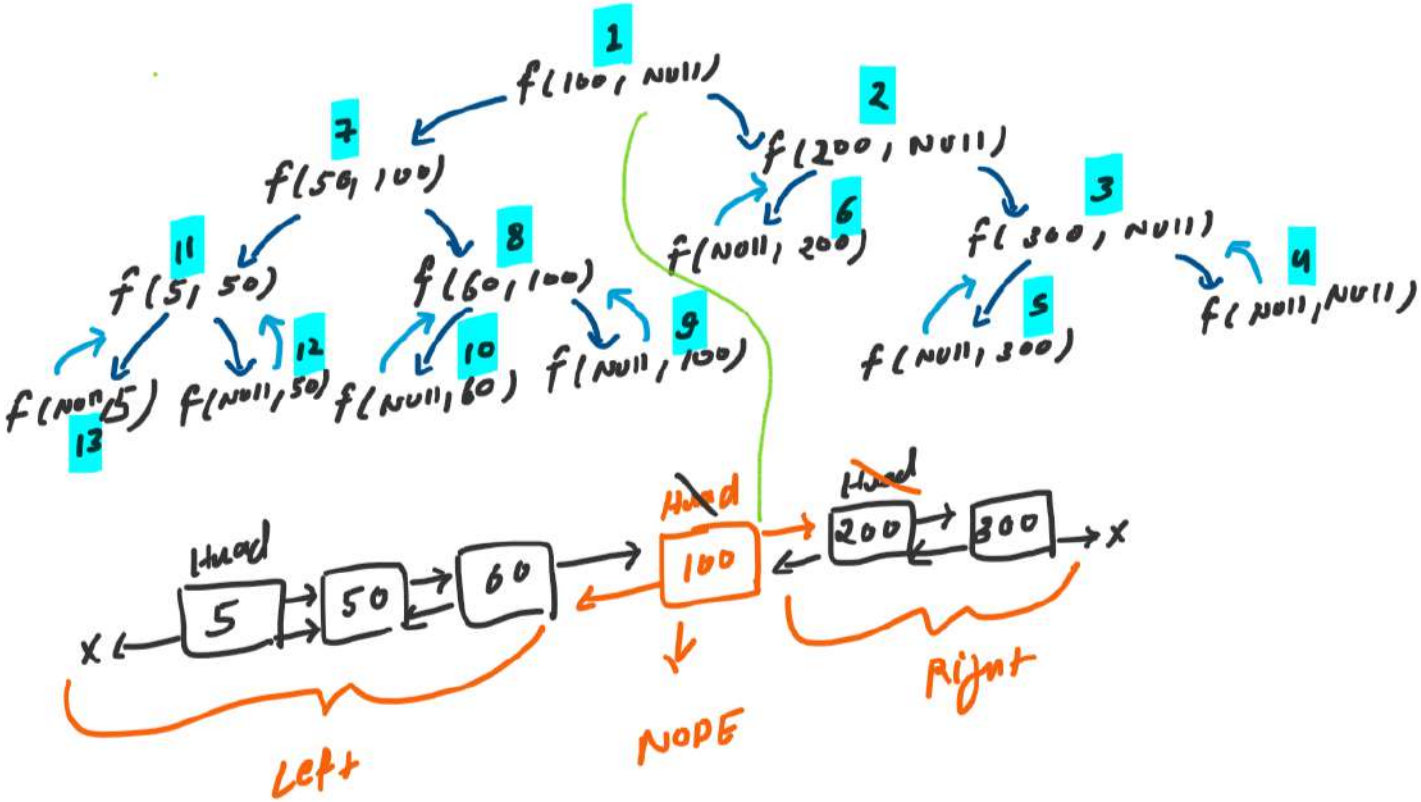

Node
f (Root, Head)
Left
Right
f(Root→left, head)
f(Root→right, head)


f(Null, 100)
6
f(100, Null)
1
Head
200
300
X
f(Null, 200)
5
f(200, Null)
2
Head
300
X
f(300, Null)
4
3
f(Null, 300)
f(Null, Null)
Head
300
X
Head
100
200
300
X

DRY RUN 2

EX2


Root
100
50
200
5
60
800
x
x
x
x
x
x
x


f(100, NULL)
f(50, 100)
f(200, NULL)
f(5, 50)
f(60, 100)
f(NULL, 200)
f(300, NULL)
f(NULL, 50)
f(NULL, 60)
f(NULL, 100)
f(NULL, 300)
f(NULL, NULL)
Head
5
50
60
100
200
300
x
Left
NODE
Right

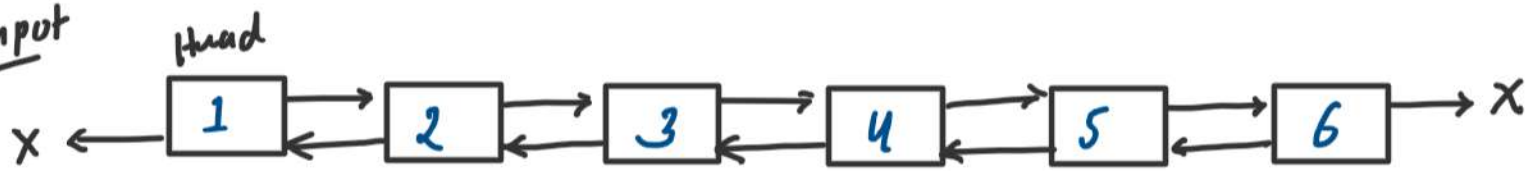
```cpp
// PROBLEM 01: Convert BST into Sorted Double Linked List

// Note: Head is queal to NULL initially
void convertBSTtoDLL(Node* root, Node* &head){
    // Base case
    if(root == NULL){
        return;
    }

    // Reverse Inorder Traversal
    // R
    convertBSTtoDLL(root->right, head);

    // N
    root->right = head;
    if(head != NULL){
        head->left = root;
    }
    // Updating the head (GALTI YANHI HO RHI THI)
    head = root;

    // L
    convertBSTtoDLL(root->left, head);
}

/*

Example 1:

INPUT:
Binary Search Tree:
60
5 200
50 100 300

OUTPUT:
Sorted Double Linked List:
5 50 60 100 200 300
*/
```

Time Complexity: *O(N)*
Space Complexity: *O(H)*

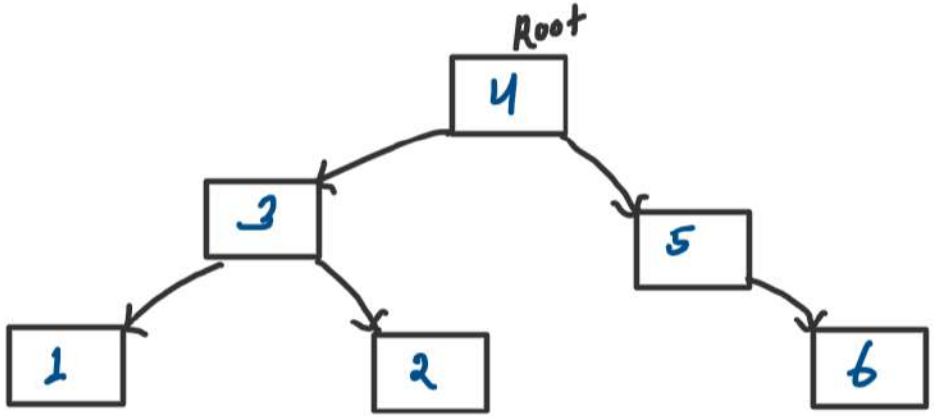
*Where **N** is number of nodes of BST and **H** is height of BST*

2. Convert Sorted Double Linked List into BST

Input



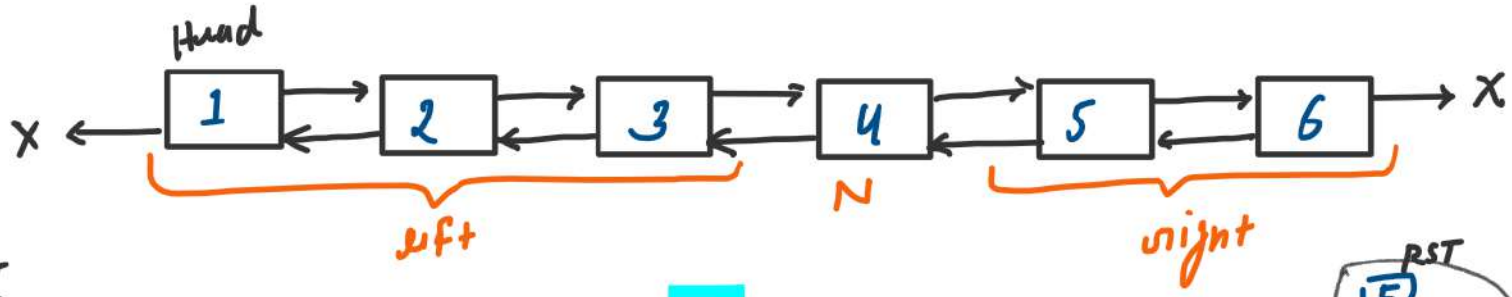
Output

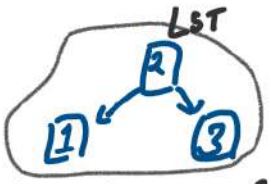


Solve using inorder traversal (LNR)

Logic Building

N = 6

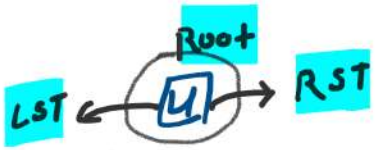




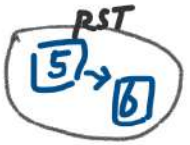
[L] Node* leftST = f(head, $\frac{N}{2}$);

Base Case

```
if(head == NULL || N<=0)
    return NULL;
```



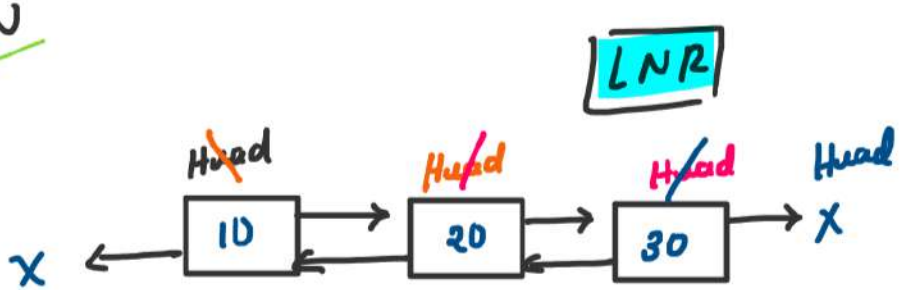
[N]
```
Node * root = head;
root -> left = leftST;
head = head -> right;
```



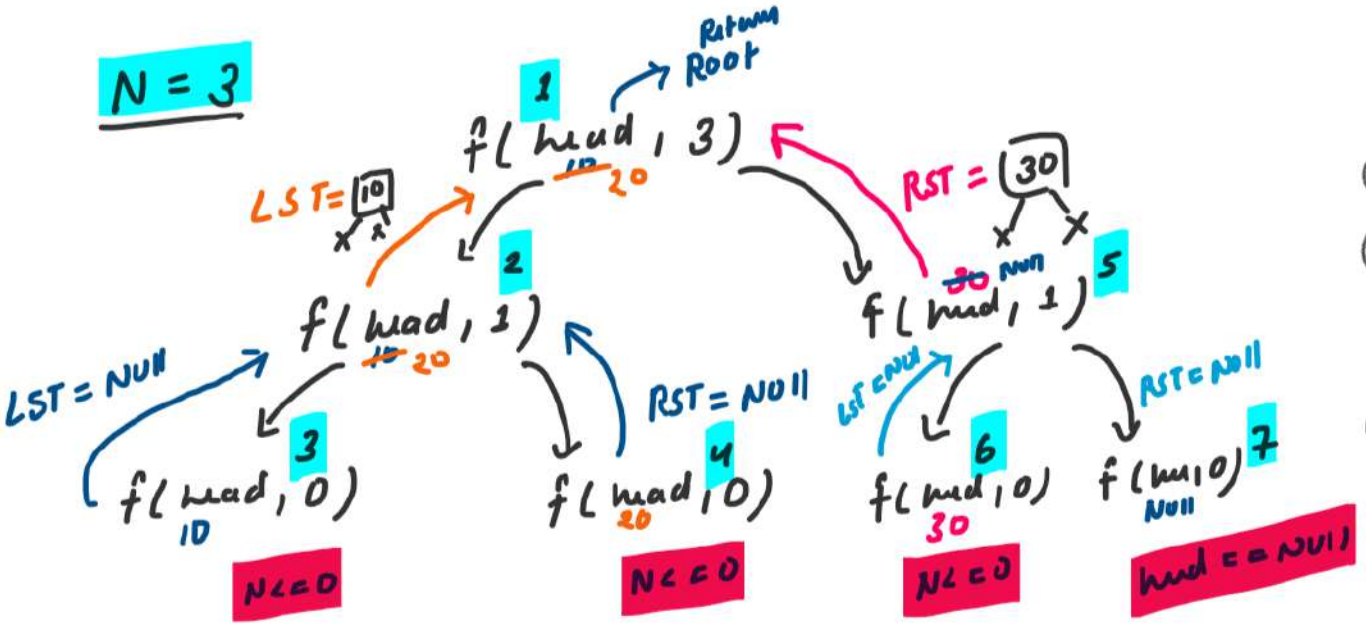
[R]
```
Node * rightST = f(head,
                   (N - N/2 - 1));
root -> right = rightST;
return Root;
```

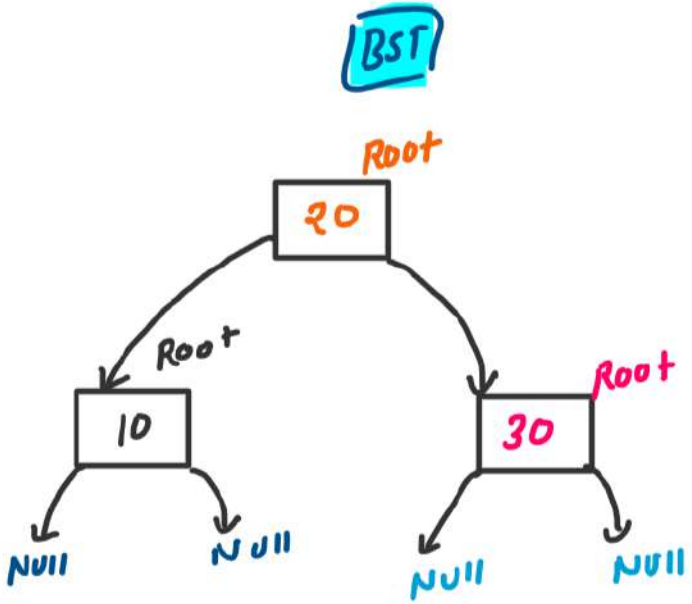
DRY RUN

EX1



N = 3



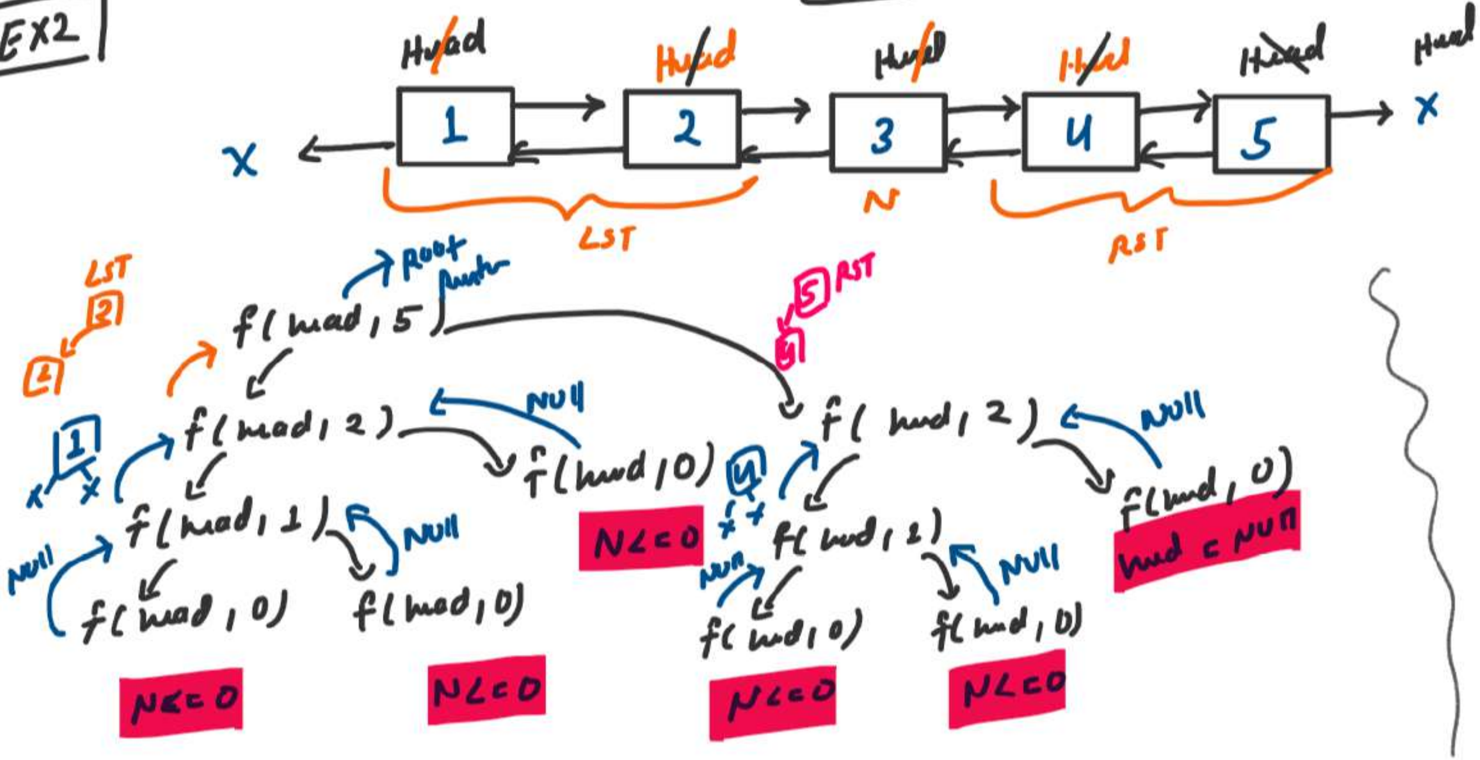


DRY RUN

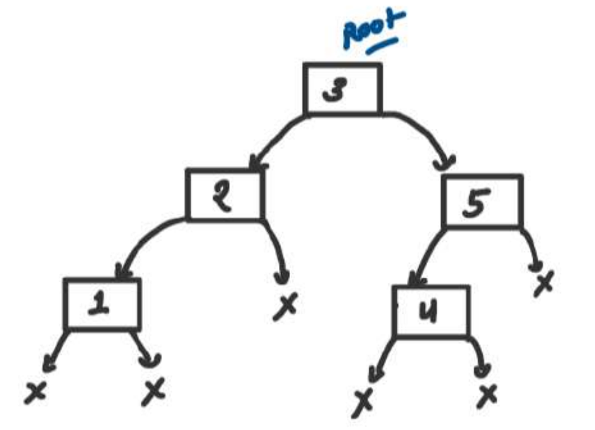
EX2

LNR

$N = 5$



BST



```cpp
// PROBLEM 02: Convert Sorted Double Linked List into BST
Node* convertDLLtoBST(Node* &head, int n){
    // Base case
    if(head == NULL || n<=0){
        return NULL;
    }

    // Inorder traversal (LNR)
    // L
    Node* leftST = convertDLLtoBST(head, n/2);

    // N
    Node* root = head;
    root->left = leftST;
    // Updating the head
    head = head->right;

    // R
    Node* rightST = convertDLLtoBST(head, (n-n/2-1));
    root->right = rightST;
    return root;
}

/*
Example 1:
Sorted Double Linked List:
10<->20<->30

Level Wise Order:
    20
    / \
   10  30
*/
```

Time Complexity: *O(N)*
Space Complexity: *O(log N) (due to the recursive call stack)*
Where N is number of nodes of DLL