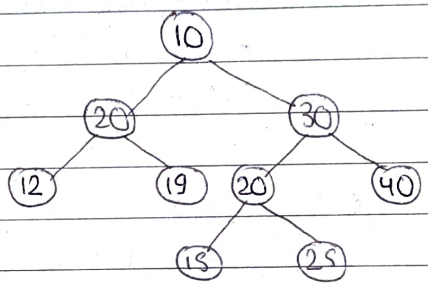
Date _____ Page _____

Day - 170

BST - 6

* Largest BST:



⇒ We have given a Binary Tree and we have to find the largest BST.

⇒ We have to find the largest BST and return the no. of nodes of it.

⇒ In the simplest approach, we can check at every node that it is BST or not, if yes then we count the no. of nodes.

⇒ This approach will take — $O(n^2)$.

⇒ So, in the above approach, we are checking every node for BST.

⇒ So, if we know that a node is following BST properties then it's ~~subtree~~

Date _____ Page _____

child are also following BST properties.
=1 So, we will use this approach.

⇒ We will do that —
→ Left BST.
→ Right BST.
→ Left side → max < root → data
→ Right side → min > root → data.
→ Size: Left side + Right side + 1

⇒ So, we will require following info —
① BST → yes or no.
② Size.
③ max
④ min

⇒ Now, we start checking & comes to (12) node then —
BST: 1 (Every leaf node is BST)
Size: 1
max: 12
min: 12
⇒ This node will pass this info to the parent node.

Code

Date _____ Page _____

```
class Box{
    public:
    bool BST;
    int size, min, max;
    Box( int data){
        BST = 1;
        size = 1;
        min = data;
        max = data;
    }
}

int largestBST (Node * root){
    int totalSize = 0;
    find( root, totalSize);
    return totalSize;
}

Box * ~~void~~ find( Node * root, int & totalsize){
    if( !root -> left && !root -> right){
        return new Box(root -> data);
    }
    else if (!root -> left && root -> right){
        Box *head = find(root -> right,
                totalsize);
        if (head -> BST && head -> min
                > root -> data){
            head -> size ++;
```

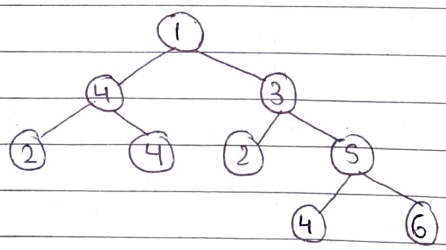
Date _____ Page _____

```
        head->min = root->data;
        TotalSize = max(totalsize, head->size);
        return head;
    } else{
        head->BST = 0; return head;
    }
}
else if(root->left && !root->right){
    Box *head = find(root->left, totalSize);
    if( head->BST && head->max <
            root->data ){
        head->size++;
        head->max = root->data;
        totalsize = max(totalSize, head->size);
        return head;
    } else{
        head->BST = 0;
        return head;
    }
} else{
    Box *leftHead = find(root->left, totalSize);
    Box * rightHead = find(root->right, totalSize);
    if( leftHead->BST && rightHead->BST
    && leftHead->max <root->data &&
    rightHead->min >root->data){
        Box *head = new Box(root->data);
        head->size += leftHead->size +
                rightHead->size;
```

Date _____ Page _____

```
        head->min = leftHead->min;
        head->max = rightHead->max;
        totalSize = max(totalSize, head->size);
        return head;
    }
    else {
        leftHead->BST = 0;
        return leftHead;
    }
  }
}
```

* Maximum Sum BST in Binary Tree:



⇒ We have to return max. sum BST from this BT.

⇒ We will use the previous approach to solve the problem.