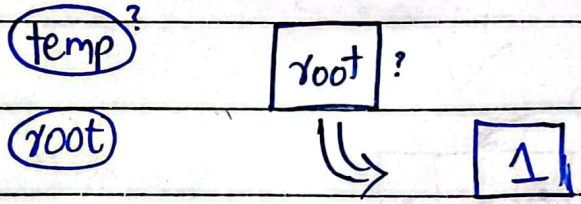
M T W T F S **DSA LAB** Date: Dec / 6 /20 23

Pre-order tree traversal:

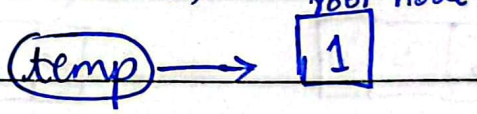
```
int main()
{
  struct Node* root = newNode(1);
```

(The newNode function is called)

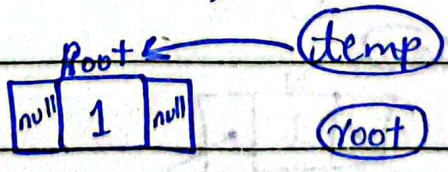
```
Node* newNode(int data)
{ Node* temp= new Node;
```




```
temp->data = data;
```

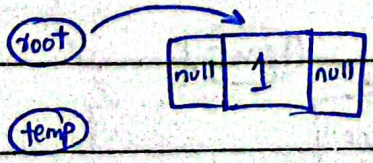



```
temp->left = temp->right = NULL;
return temp; }
```

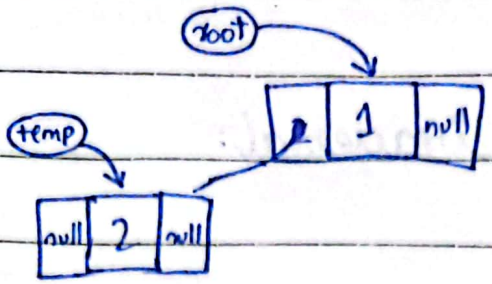


(again going to main function)

```
root->left = newNode(2);
```

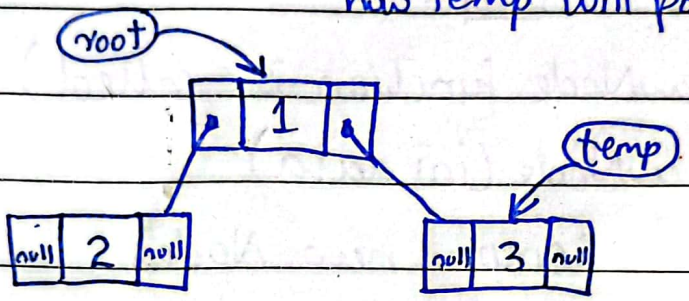


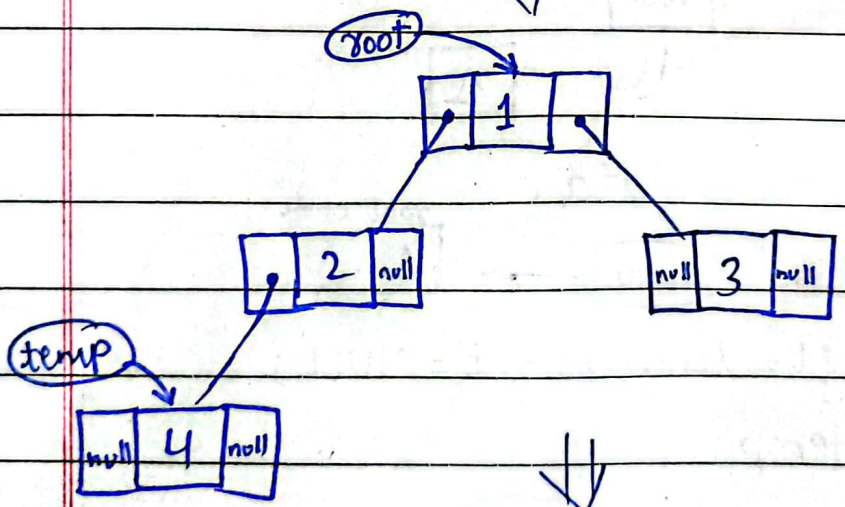
(again newNode function called)

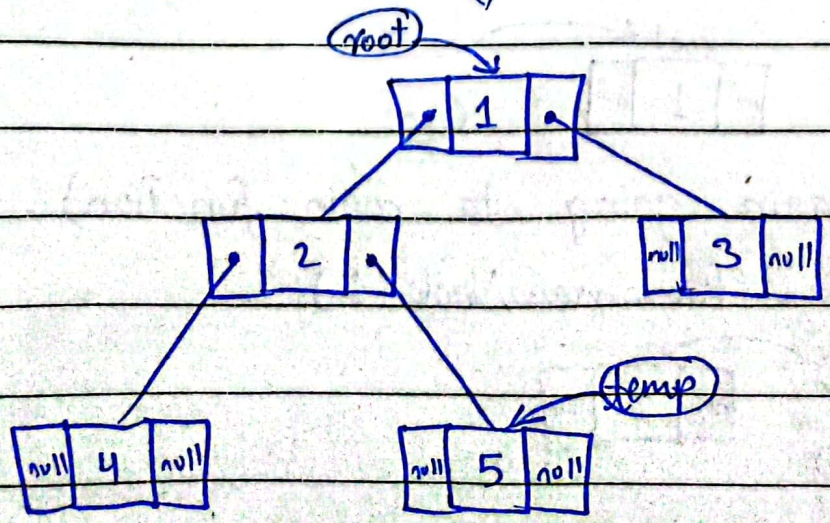


- root → right = newNode(3);

(same procedure repeated, now temp will point to right)







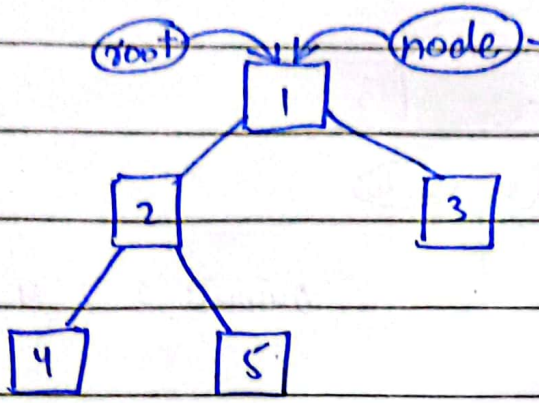
Date: / /20

- printPreorder(root);

(now printPreorder function is called)

(if condition doesn't apply so...)

- cout << node -> data << " ";

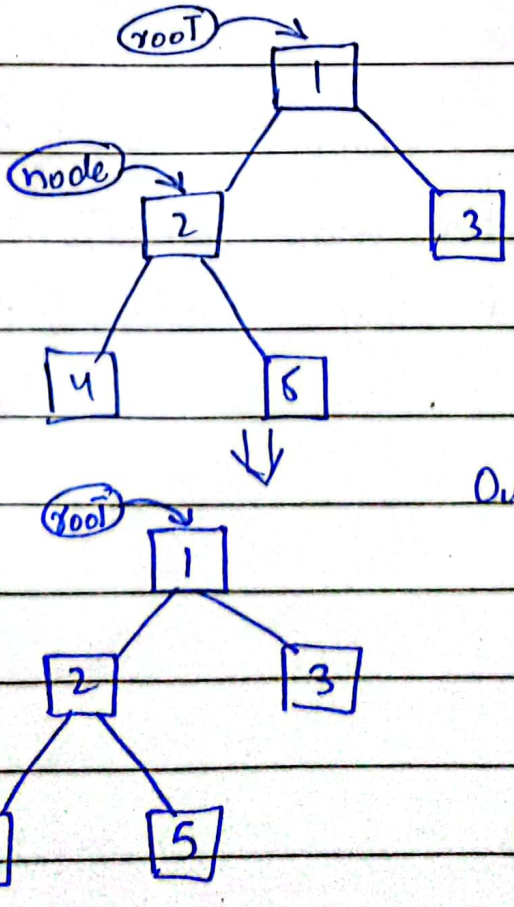


→ This pointer from [void printPreorder(struct Node* node)]

By this, node's data is printed (1).

Output: 1

- printPreorder(node->left);



(again, the whole function is called for left node)

Output: 1 2

Output: 1 2 4

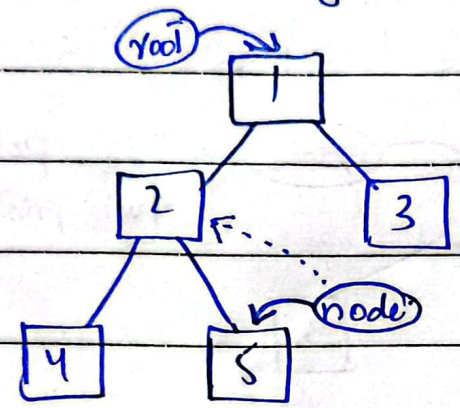
Excellent ☐ Good ☐ Need improvement ☐

M T W T F S

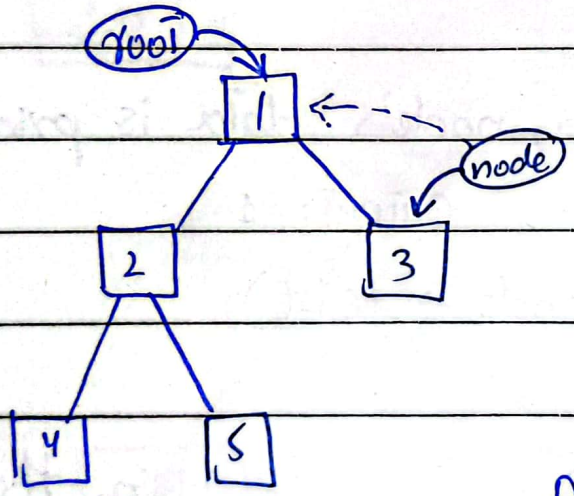
Date: ..../..../20....

Now, this node "4" doesn't has left subtree (if condition applied), so function returns.

- printPreorder(~~right~~ node->right);



Output: 1 2 4 5



(function is called again)

Output: 1 2 4 5 3

After this, the main function is terminated by return 0;.

M T W T F S Date: /20

Post-order tree traversal:

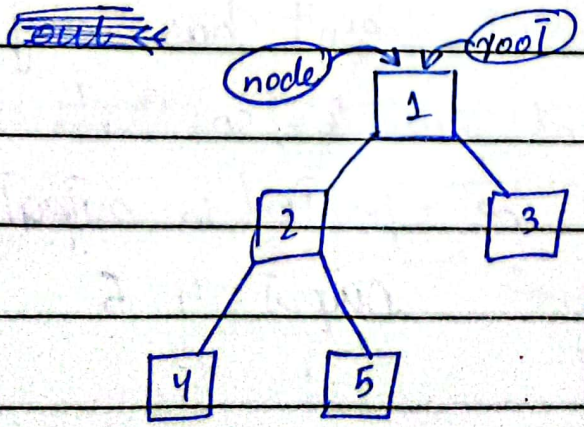
```
int main(){
struct Node* root = newNode(1);
root->left = newNode(2);
root->right = newNode(3);
root->left->left = newNode(4);
root->left->right = newNode(5);
```

(Tree is created by same procedure as previous code).

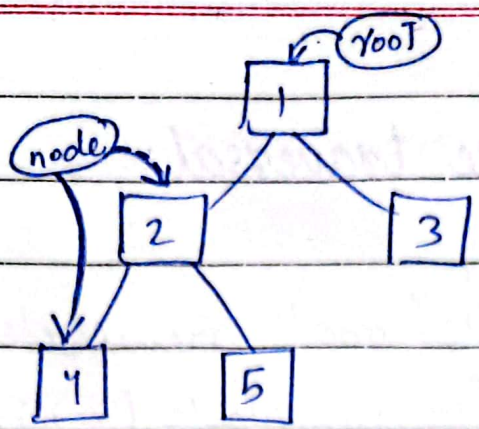
```
printPostorder(root);
```

(printPostorder function is called)

if-condition is not applicable so function proceeds to next line.




```
printPostorder(node->left);
```



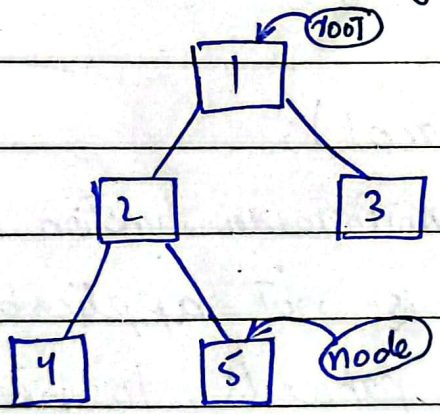
(whole function called again for each node

Now "4" node doesn't has left & right subtree, so its data is printed in output by

- cout << node → data << " ";

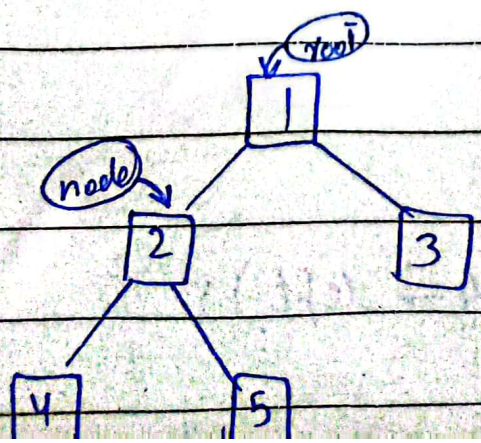
Output: 4

- printPostorder(node → right);



Now, this "5" doesn't has any left or right subtree, so; its data will be printed in output.

Output: 4 5

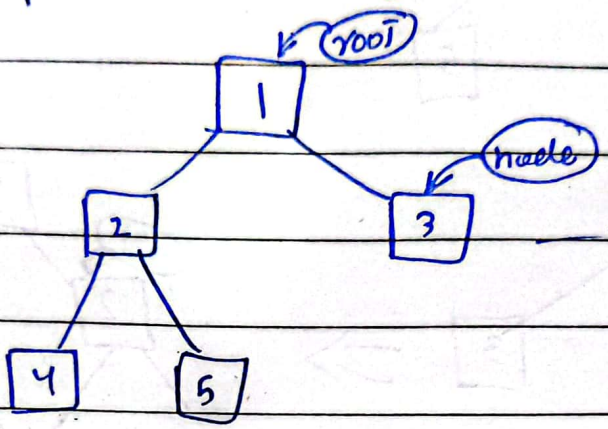


Output: 4 5 2

Again, function calls:

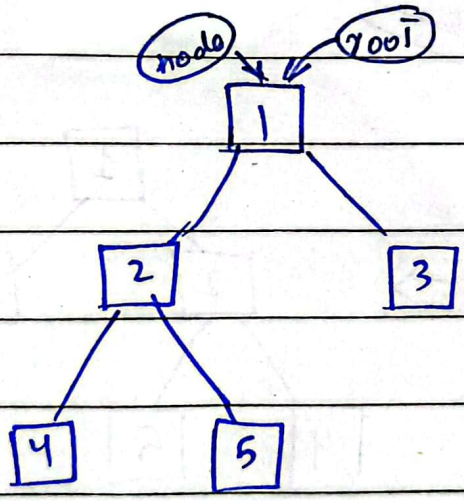
- printPostorder(node->right);

Now, in function, node pointer points to right of root.



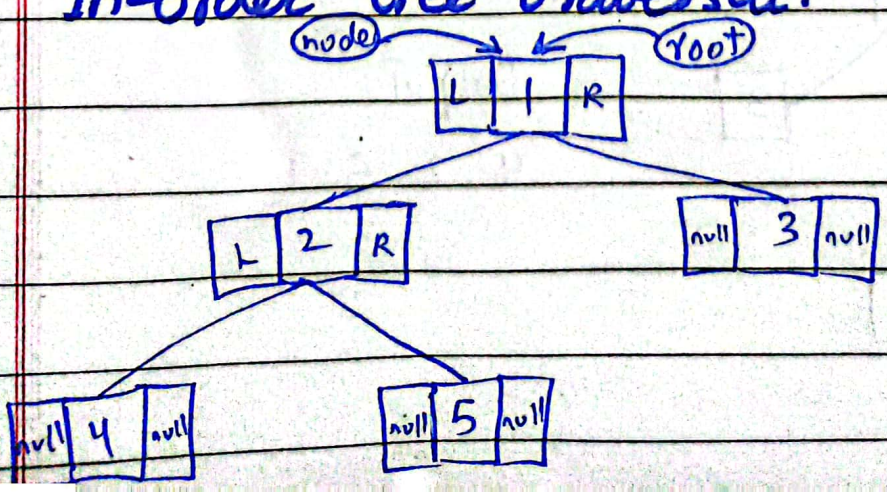
→ (doesn't has left or right subtree)

Output: 4 5 2 3



Output: 4 5 2 3 1.

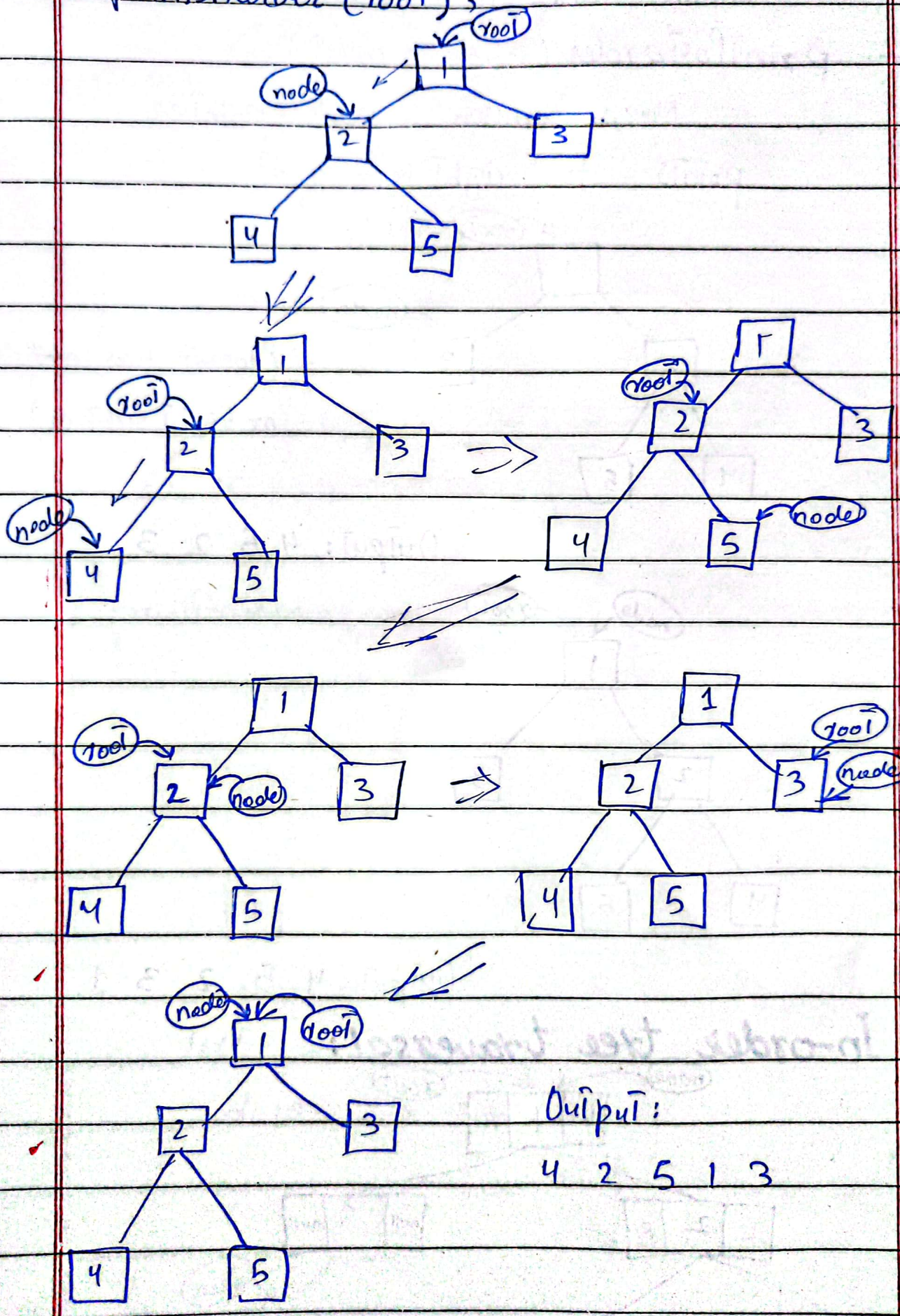
- **In-order tree traversal:**



M T W T F S

Date: ... / ... /20...

- PrintInorder (root);

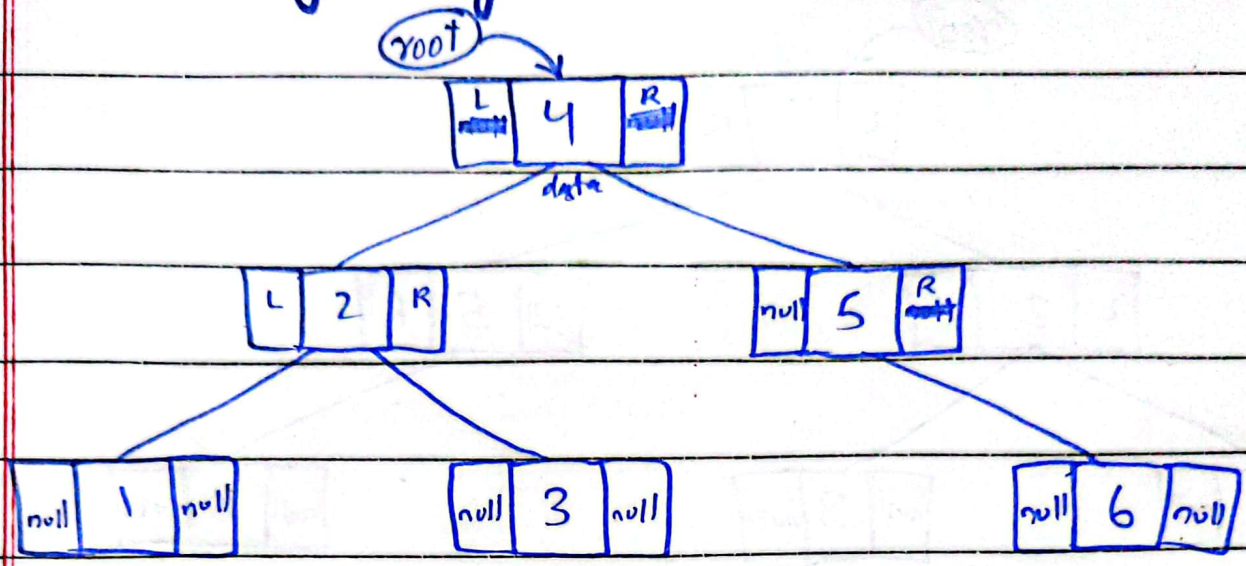


Output:

4 2 5 1 3

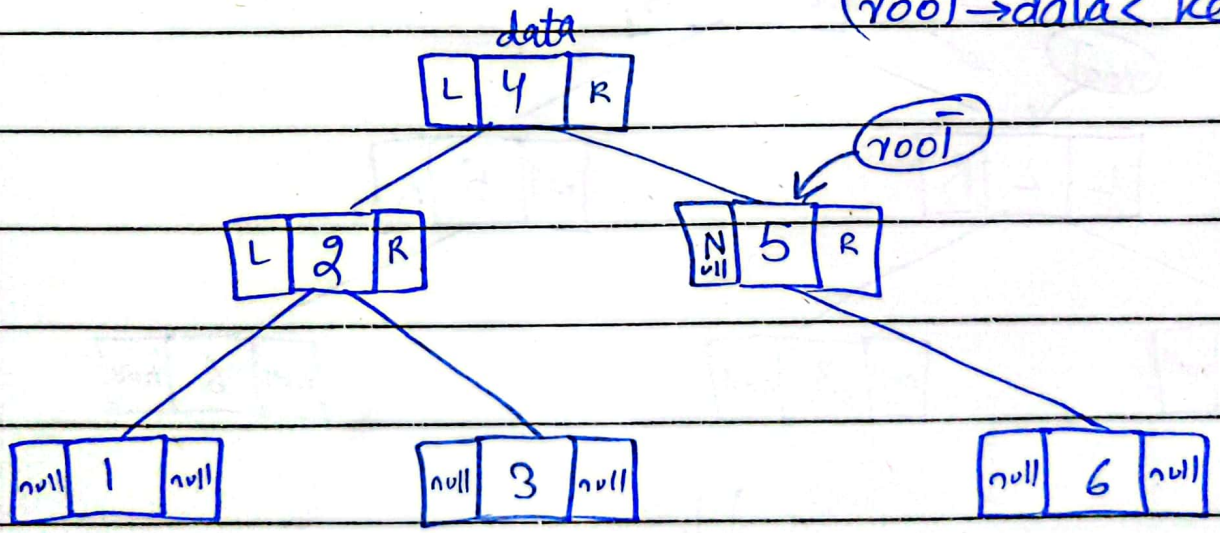
Date: ...../...../20.....

- **Searching any node in BST:**



Key = 5 (value To be searched.)

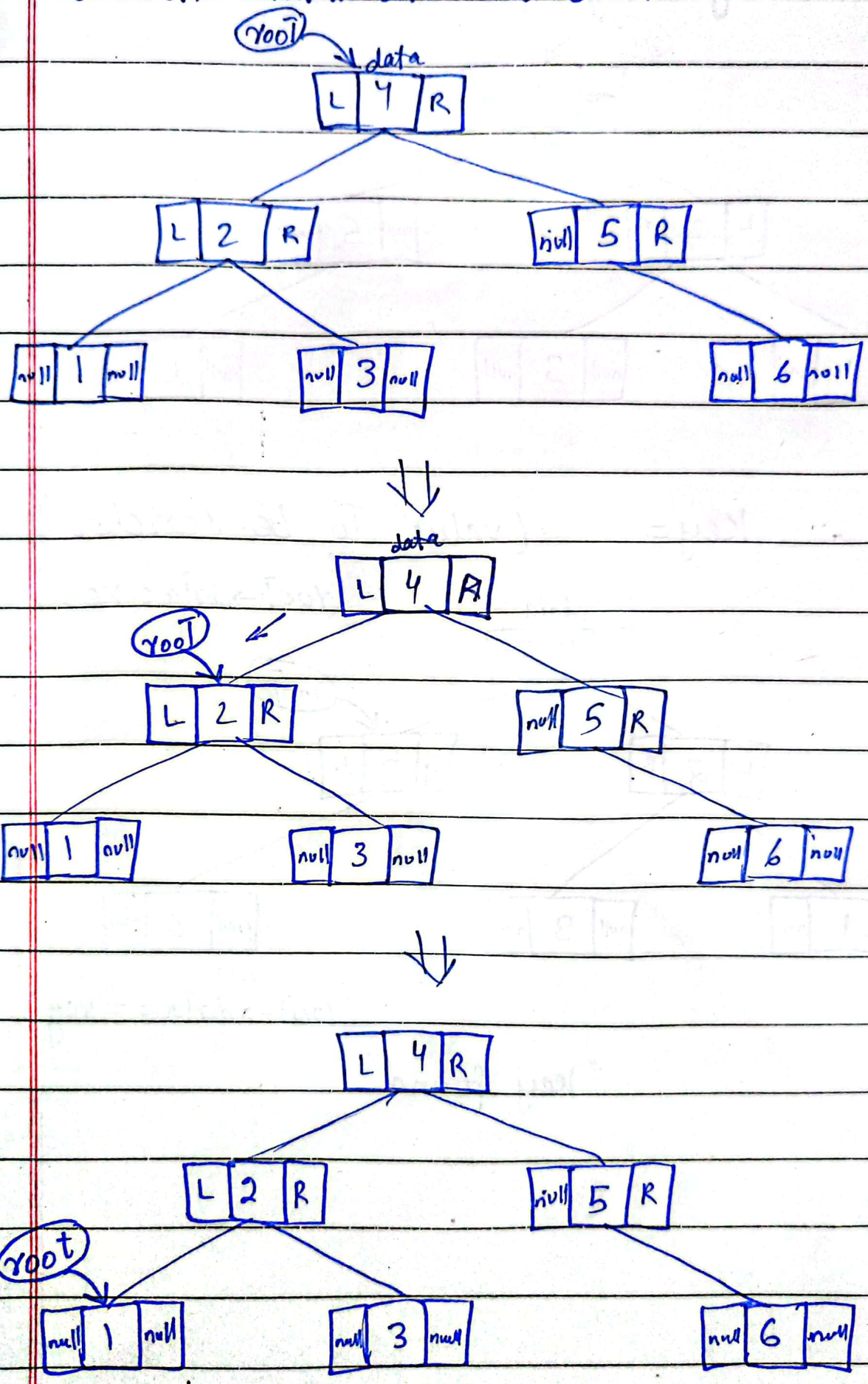
(root→data < key)



(root→data == key)

"key found"

Search minimum value in BST:



"Min-value found"