Submitted by:-

Mubeshra Shafique

Roll no.:- SP22-BCS-120

Section:- B

Subject:-

Data Structure (Lab)

Submitted to:-

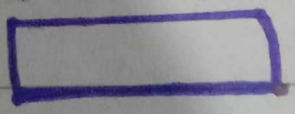
Mam Yasmeen Jana

Traversal in B.T

1- Inorder:-

→ First of all go to main function

struct node * root = creatNode(1);



→ Now go to the creatNode function

struct node *creatNode (int val) {

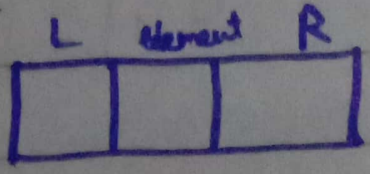
struct node *Node = new struct node.

→ move on struct node part

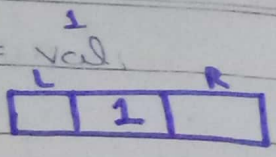
struct node* left;

struct node* right;

int element;



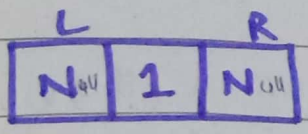
Node → element = val;



Node → left = NULL;

Node → right = NULL;

return Node;



root

return Node;

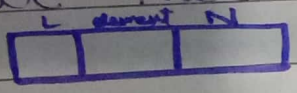
→ again go to the main function.

root → left = CreateNode (2);

→ move to the CreateNode function.

struct node *CreateNode (int val) {

struct node * Node = new struct node;

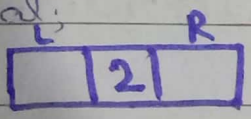


struct node *left;

struct node *right;

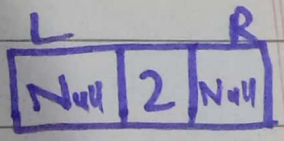
int element;

Node → element = val;



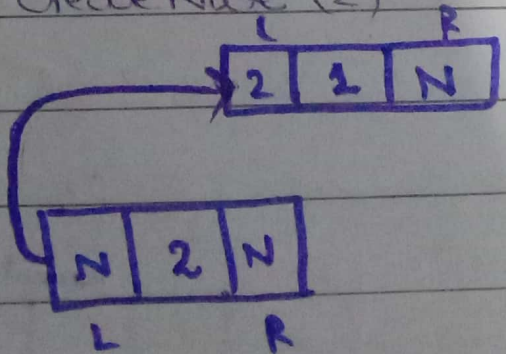
Node → left = NULL;

Node → right = NULL;

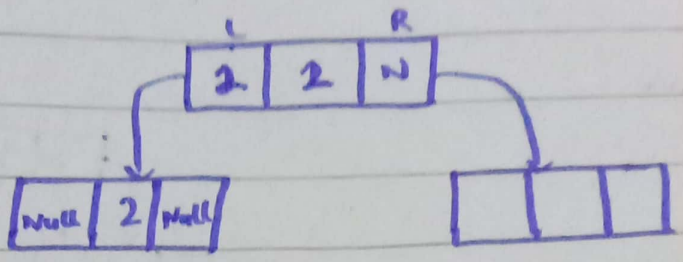


return Node;

root → left = CreateNode (2)

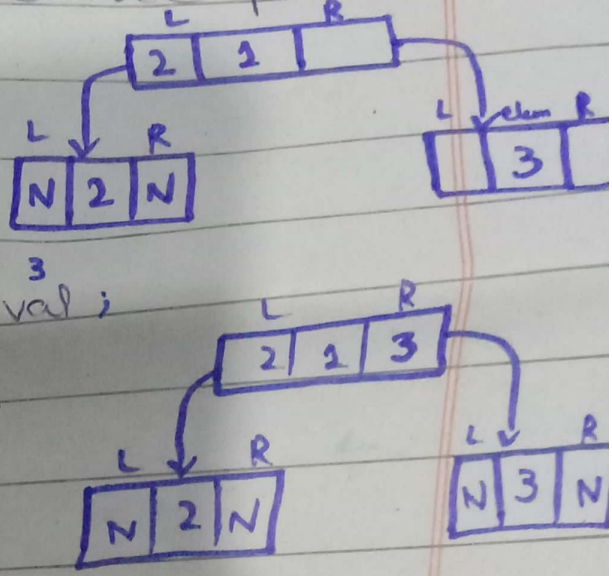


→ move to the main function
root->right = createNode(3);

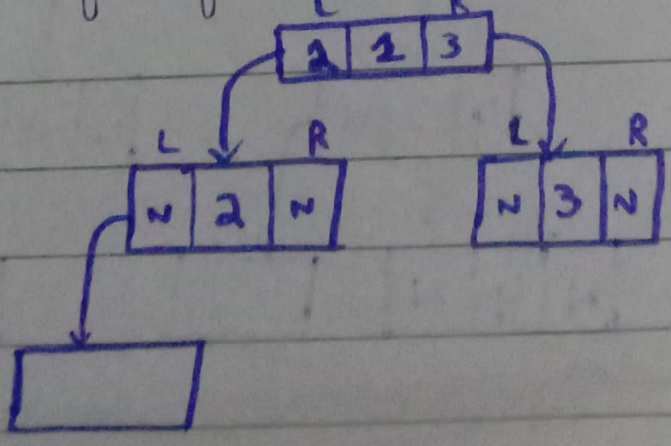


→ move to the createNode func.
Struct node *Node = new struct node;

→ move to the struct node part.
int element,
struct node * left,
struct node * right;
Node → element = val;
Node → left = Null;
Node → right = NULL;
return Node;



→ again move to the main function
root → left → left = createNode(4);



→ move to the createNode function

struct node *Node (int x);

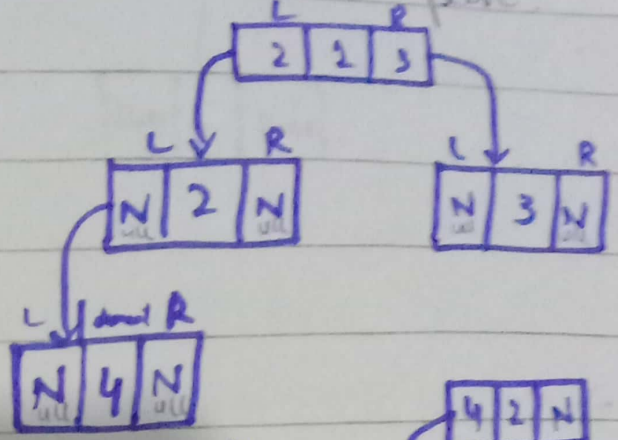
struct node *Node = new struct node;

→ move to the struct node part

int element

struct node *left;

struct node *right;

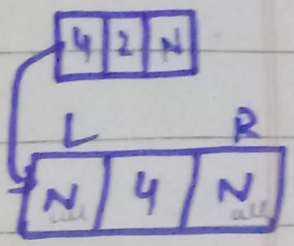


Node → element = val;

Node → left = NULL;

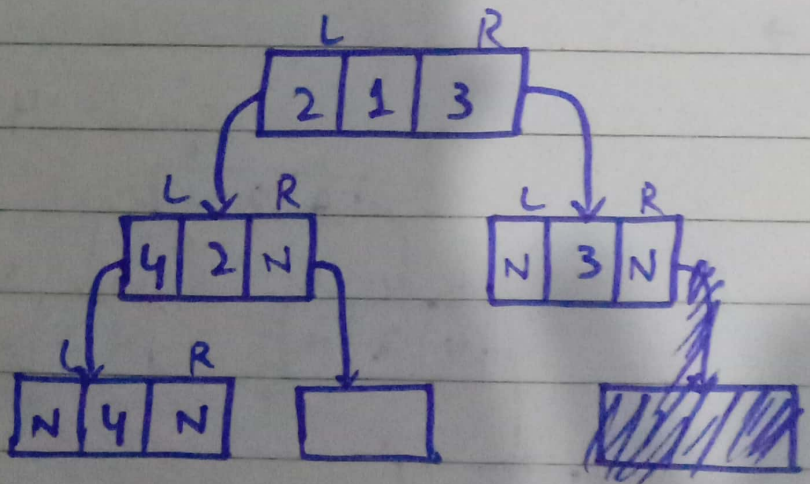
Node → right = NULL;

return node;



→ again move to the main function

root → left → right = createNode(5);



→ move to the createNode function

struct node * createNode(int val){

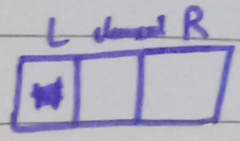
struct node *Node = new struct node;

move to the struct node part;

int element;

struct node *left;

struct node *right;

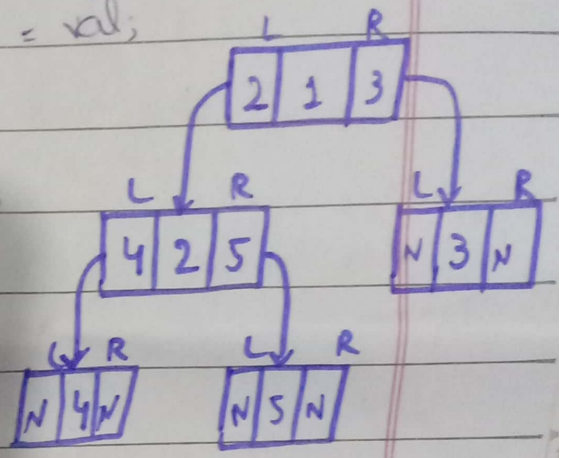


Node → element = val;

Node → left = NULL;

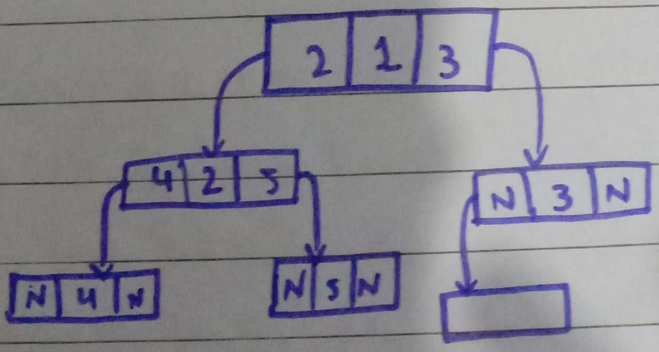
Node → right = NULL;

return node;



→ again move to the main function.

root → right → left = createNode(6);



→ move to the createNode function.

struct node *createNode(int val){

struct node *Node = new struct node;

→ move to the struct node part.

int element;

struct node * left;

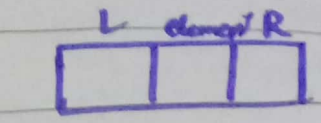
struct node * right;

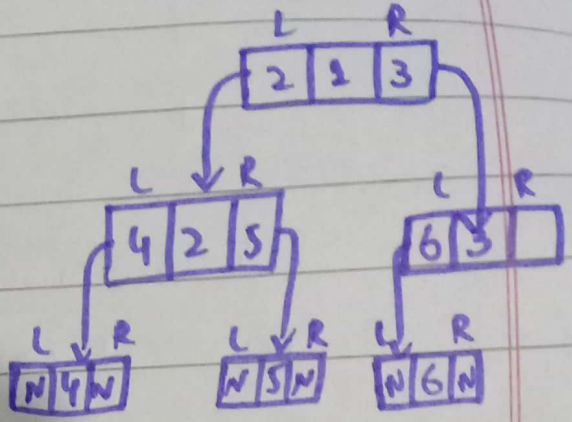
Node→element = val;

Node→left = NULL;

Node→right = NULL;

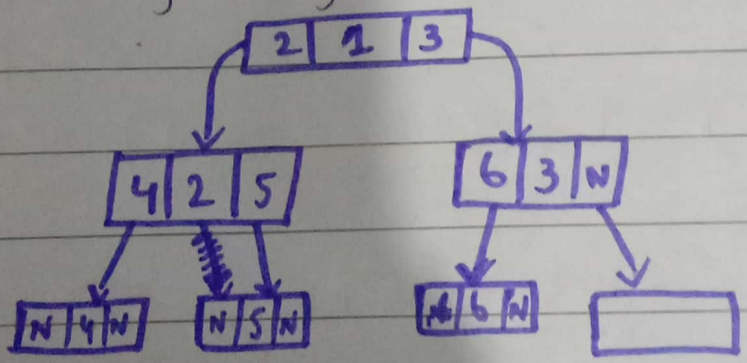
return node;





→ again move to the main function.

root→right→right = createNode(7);



→ move to the createNode part.

struct node *createNode(int val){

struct node *Node = new struct node;

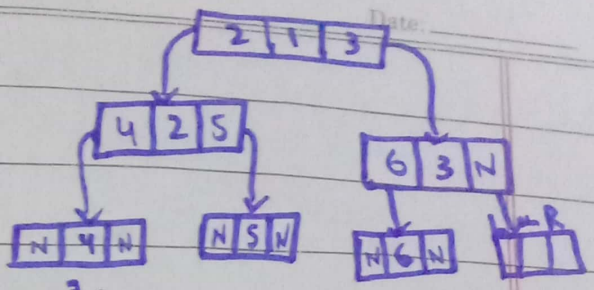
→ move to the struct node part

int element

struct node *left;

Day: _____ Date: _____

Struct node *right

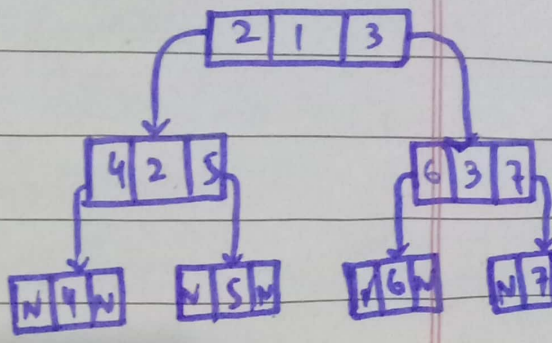


Node → element = val;

Node → left = NULL;

Node → right = NULL;

return Node;



→ again move to the main function.

cout<< "The in-order traversal of the given binary tree is " →

traverseInorder (root);

→ move to the traverseInorder function.

void traverseInorder (struct node *root) {

if (root == NULL) { → False

return;

}

traverseInorder (root → left); 2

→ Call the traverseInorder function.

if (root = = NULL) { → Now root become 2

return; → False

traverseInorder (root → left); 4

→ again call the traverseInorder function

if (root == NULL) → True

return; → return chalg gy

cout<< root→element; 4

cout<< root→element. 2

traverseInorder(root→right); 2 5

→ again call to the traverseInorder

if (root == NULL) → False

return;

cout<< root→element ; 5

→ again move to the main root

cout<< root→element; 1

traverseInorder(root→right); 1 3

→ again call the traversInorder func.

if (root == NULL) → False 6

return;

traverseInorder (root→left); 6

→ again call the traverseInorder func.

if (root == NULL) → True

return; skip the neechy wali lines.

cout<< root→elemet; 6

traverseInorder (root→right); 3 7

cout<< root→element; 3

traverseInorder (root→right); 3 7

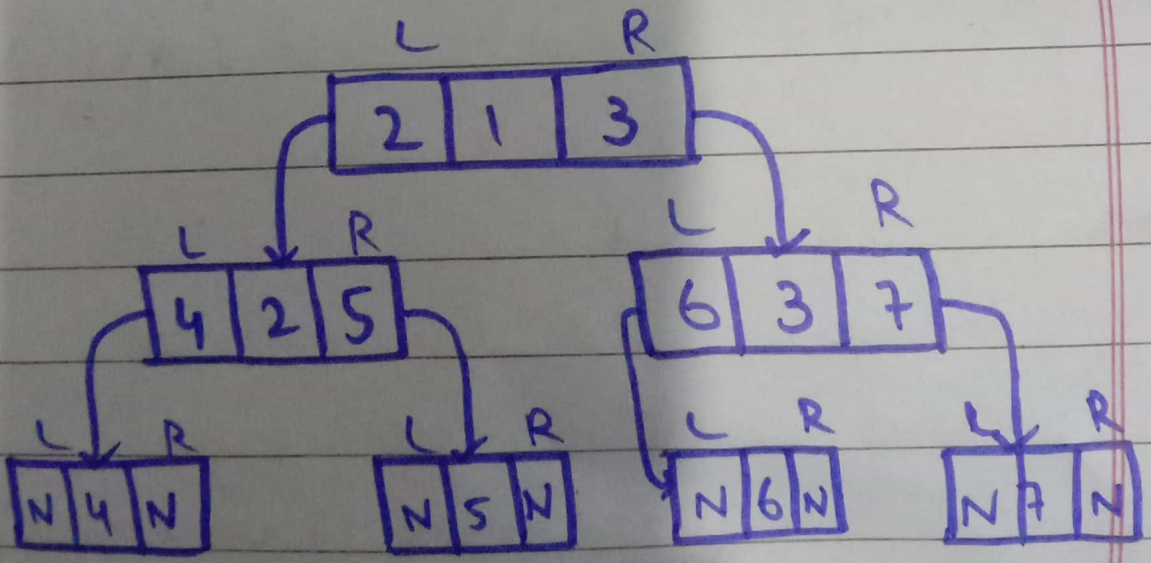
→ again call the traverseInorder func.
if (root == NULL) → False
return;
traverseInorder(root → left)

→ again call the traverseInorder func.
if (root == NULL) → True
return ; chaly ga cab or neechy wali lines skip

Cout<< root → element; 7

4 2 5 1 6 3 7

2- Preorder.



→ go to the main function
print preorder (root);
→ go to the print order function

if (root^{1} == NULL) → False
return;

~~print preorder~~

cout << node → data;

print preorder(root → left);

print preorder (root → right);

print 1

print preorder (root → left) → 2

→ go to if condition

if (root^{2} == NULL) → False

return;

cout << node → data;

print 2

print preorder (root → left) → 4

cout << node → data.

print 4

→ go back to the root 2 and go to the right.

print preorder (root → right) → 5

cout << node → data;

print 5

→ go back to the root 1 and go to the right.

print preorder (root → right) → 3

called node → data

print 3

→ go back to the left side of the root 3.

print preorder (root → left) → 6

print 6

→ go to the right side of root 3

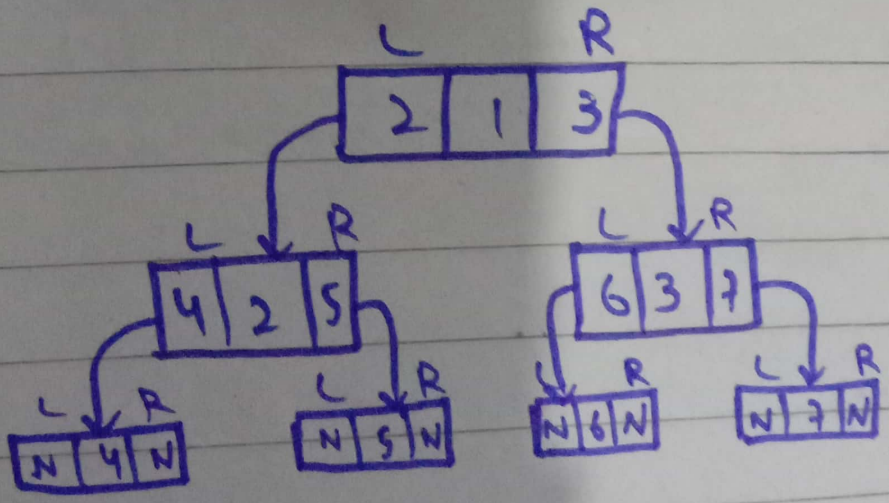
~~go to the right~~ :

print preorder (root → right) → 7

print 7

1 2 4 5 3 6 7

3- Postorder:-



→ go back to the main function.

print postorder (root);

→ go to the print preorder function.

if (root == NULL) → False
return.

print postorder (root → left) 2

→ again go to if condition

if (root == NULL) → False.
return.

print postorder (root → left) 4

→ So there is no child of 4
so
print 4

→ Back to the previous call
print postorder (root → right)

→ So there is no child of 5
so print 5.

→ Back to the main root 1

→ print the data of the current node.

print 2.

→ Now call. printpostorder.

printpostorder (root → right) →3

→ go to the printpostorder

printpostorder (root → left) →6

→ So there is no child of 6

print 6

Date: / / Day:

→ Back to the previous call printpostorder (root→right) →7

→ So, there is no child of the node 7.

→ Back to the original root print 1.

→ The final output of the postorder traversal is.

4 5 6 7 3 1

Insertion and deletion of Binary search tree.

→ go to the main function

Node * root = null,