

DSA ASSIGNMENT – 4

21BCE7371 RADHA KRISHNA GARG

Implementation in C

INPUT

```
#include <stdio.h>

#include <stdlib.h>

struct node {
    int value;
    struct node* left;
    struct node* right;
};

// Inorder traversal
void InOrder(struct node* root) {
    if (root == NULL) return;
    InOrder(root->left);
    printf("%d ", root->value);
    InOrder(root->right);
}

// PreOrder traversal
void PreOrder(struct node* root) {
    if (root == NULL) return;
    printf("%d ", root->value);
    PreOrder(root->left);
    PreOrder(root->right);
}
```

```

}

// PostOrder traversal
void PostOrder(struct node* root) {
    if (root == NULL) return;
    PostOrder(root->left);
    PostOrder(root->right);
    printf("%d ", root->value);
}

// Create a new Node
struct node* createNode(int value) {
    struct node* newNode = malloc(sizeof(struct node));
    newNode->value = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

int main() {
    struct node* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);
    root->right->left = createNode(6);
    root->right->right = createNode(7);
    printf("Inorder traversal:\t");
    InOrder(root);

```

```

printf("\PreOrder traversal:\t");
PreOrder(root);
printf("\nPostOrder traversal:\t");
PostOrder(root);
}

```

main.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  struct node {
4      int value;
5      struct node* left;
6      struct node* right;
7  };
8  // Inorder traversal
9  void InOrder(struct node* root) {
10     if (root == NULL) return;
11     InOrder(root->left);
12     printf("%d ", root->value);
13     InOrder(root->right);
14 }
15 // PreOrder traversal
16 void PreOrder(struct node* root) {
17     if (root == NULL) return;
18     printf("%d ", root->value);
19     PreOrder(root->left);
20     PreOrder(root->right);
21 }
22 // PostOrder traversal
23 void PostOrder(struct node* root) {
24     if (root == NULL) return;
25     PostOrder(root->left);
26     PostOrder(root->right);
27     printf("%d ", root->value);

```

main.c

```
25     postorder(root->left);
26     PostOrder(root->right);
27     printf("%d ", root->value);
28 }
29 // Create a new Node
30 struct node* createNode(int value) {
31     struct node* newNode = malloc(sizeof(struct node));
32     newNode->value = value;
33     newNode->left = NULL;
34     newNode->right = NULL;
35     return newNode;
36 }
37 int main() {
38     struct node* root = createNode(1);
39     root->left = createNode(2);
40     root->right = createNode(3);
41     root->left->left = createNode(4);
42     root->left->right = createNode(5);
43     root->right->left = createNode(6);
44     root->right->right = createNode(7);
45     printf("Inorder traversal:\t");
46     InOrder(root);
47     printf("\PreOrder traversal:\t");
48     PreOrder(root);
49     printf("\nPostOrder traversal:\t");
50     PostOrder(root);
51 }
```

OUTPUT

Output

/tmp/t7C6GPU04D.o

```
Inorder traversal:  4 2 5 1 6 3 7 PreOrder traversal:  1 2 4 5 3 6 7
PostOrder traversal:  4 5 2 6 7 3 1
```

Implementation in Python

```
from asyncio import Queue

class Node:
    def __init__(self, item):
        self.left = None
        self.right = None
        self.val = item

# creating a tree data structure
def inorder(root):
    #checking if the root is null or not
    if root:
        inorder(root.left)
    # recursively calling left subtree
    print(str(root.val) + " ", end = '')
    inorder(root.right)
    # recursively calling right subtree
def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(str(root.val) + " ", end = '')
def preorder(root):
    if root:
        print(str(root.val) + " ", end = '')
        preorder(root.left)
        preorder(root.right)
def levelOrder(root):
    queue = list()
    queue.append(root)
    while len(queue):
        current = queue[0]
        queue = queue[1: ]

    print(str(current.val) + " ", end = "")
    if current.left:
        queue.append(current.left)
    if current.right:
        queue.append(current.right)
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
```

```

root.left.right = Node(5)
root.right.left = Node(6)
root.right.right = Node(7)
print("\nLevelOrder traversal:\t", end = " ")
levelOrder(root)
print("\nInorder traversal:\t", end = " ")
inorder(root)
print("\nPreorder traversal:\t", end = " ")
preorder(root)
print("\nPostorder traversal:\t", end = " ")
postorder(root)

```

Implementation in JAVA

```

// Tree traversal in Java
class Node
{ int item;
  Node left, right;
  public Node(int key)
  { item = key;
    left = right = null;
  } }
class BinaryTree {
  // Root of Binary Tree
  Node root;
  BinaryTree() {
    root = null;
  }
  void postorder(Node node)
  { if (node == null)
    return;
    // Traverse left
    postorder(node.left);
    // Traverse right
    postorder(node.right);
    // Traverse root
    System.out.print(node.item + "->");
  }
  void inorder(Node node)
  {
    if (node == null)
    return;
    // Traverse left
    inorder(node.left);
    // Traverse root
    System.out.print(node.item + "->");
    // Traverse right

```

```

inorder(node.right); }
void preorder(Node node)
{
if (node == null) return;
// Traverse root
System.out.print(node.item + "->");
// Traverse left
preorder(node.left);
// Traverse right
preorder(node.right); }
public static void main(String[] args) { BinaryTree tree = new BinaryTree();
    tree.root = new Node(1);
    tree.root.left = new Node(12);
    tree.root.right = new Node(9);

    tree.root.left.left = new Node(5);
    tree.root.left.right = new Node(6);
    System.out.println("Inorder traversal");
    tree.inorder(tree.root);
    System.out.println("\nPreorder traversal ");
    tree.preorder(tree.root);
    System.out.println("\nPostorder traversal");
    tree.postorder(tree.root);
} }

```

OUTPUT

```

> ▾ TERMINAL
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\krish\Documents\java> & 'C:\Program Files\Java\jdk-18.0.1\bin\java.exe' '-XX:+ShowCodeDetails
sh\AppData\Roaming\Code\User\workspaceStorage\2c70a5e9ead433afa7c9efeb6ea4f1c5\redhat.java\jdt_ws\java_d26
Inorder traversal
5->12->6->1->9->
Preorder traversal
1->12->5->6->9->
Postorder traversal
5->6->12->9->1->
PS C:\Users\krish\Documents\java>

```