Jashore University of Science and Technology

Department of Computer Science and Engineering

Course Code: **CSE 1201**

Course Title: **Data Structures**

Assignment On

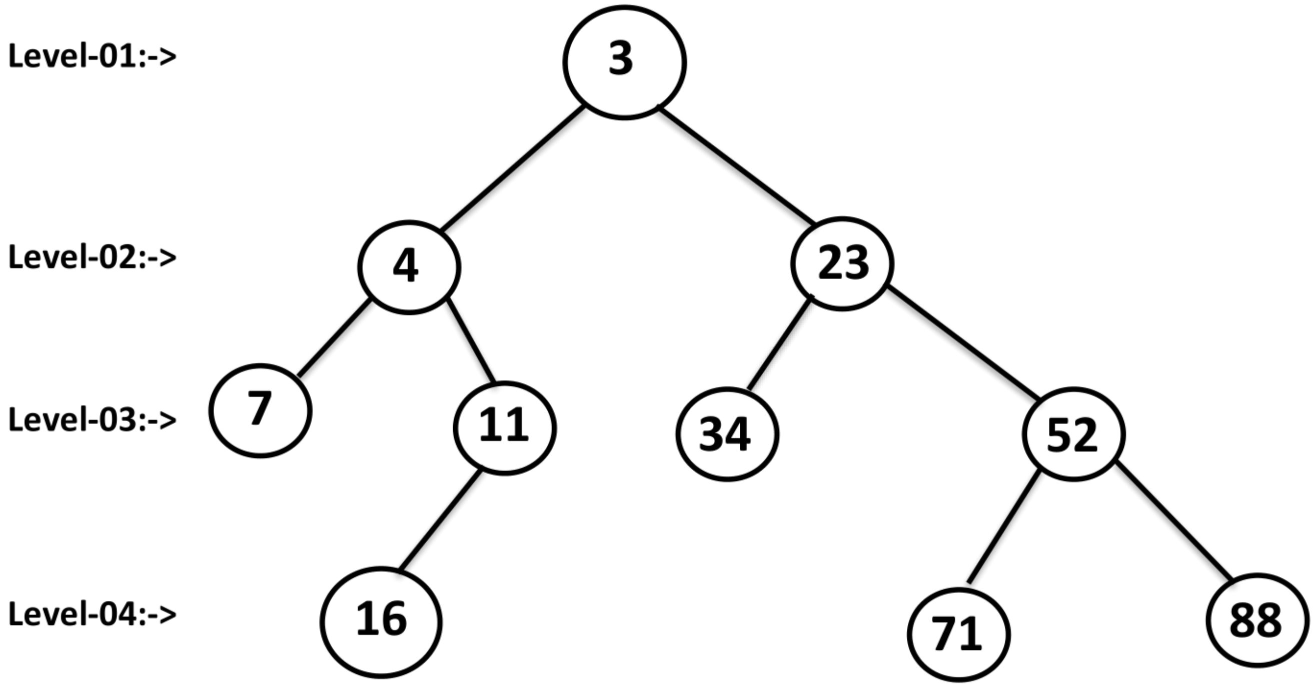
Binary Tree Traversals – Preorder, Inorder and Postorder

Submitted to	Submitted by
Dr. Md. Alam Hossain Associate Professor Dept. of Computer Science and Engineering Jashore University of Science and Technology	Mirza Mohibul Hasan Student ID: 190119 1st year, 2nd semester Dept. of Computer Science and Engineering Jashore University of Science and Technology

Date of Performance: 02.06.2021

Date of Submission: 05.06.2021

Binary Tree:



Preorder : 3 4 7 11 16 23 34 52 71 88

Inorder : 7 4 16 11 3 34 23 71 52 88

Postorder : 7 16 11 4 34 71 88 52 23 3

Algorithm for Preorder traversal:

1. Visit the root.
2. Traverse the left subtree.
3. Traverse the right subtree.

Algorithm for Inorder traversal:

1. Traverse the left subtree.
2. Visit the root.
3. Traverse the right subtree.

Algorithm for Postorder traversal:

1. Traverse the left subtree.
2. Traverse the right subtree.
3. Visit the root.

C program for Binary Tree Traversal:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};
struct Node* createNode(int data)
{
   struct Node* Node;
   Node = (struct Node*)malloc(sizeof(struct Node));
   Node->data = data;
   Node->left = NULL;
   Node->right = NULL;
   return (Node);
}
void PreOrder(struct Node* Node)
{
   if (Node == NULL)
      {
         return;
      }

   printf("%d ",Node->data);
   PreOrder(Node->left);
   PreOrder(Node->right);
}
void InOrder(struct Node* Node)
{
   if (Node == NULL)
      {
```

```
      return;
    }
  InOrder(Node->left);
  printf("%d ",Node->data);
  InOrder(Node->right);
}
void PostOrder(struct Node* Node)
{
  if (Node == NULL)
    {
      return;
    }

  PostOrder(Node->left);
  PostOrder(Node->right);
  printf("%d ",Node->data);
}
int main()
{
  struct Node* root;

  root = createNode(3);
  root->left = createNode(4);
  root->right = createNode(23);
  root->left->left = createNode(7);
  root->left->right = createNode(11);
  root->right->left = createNode(34);
  root->right->right = createNode(52);
  root->left->right->left = createNode(16);
  root->right->right->left = createNode(71);
  root->right->right->right = createNode(88);

  printf("Preorder Traversal: ");
  PreOrder(root);
```

```
    printf("\nInorder Traversal: ");
    InOrder(root);

    printf("\nPostorder Traversal: ");
    PostOrder(root);

 return 0;
}
```