# Data Structures Odyssey: Exploring the Foundations of Computing

| | | |
|---|---|---|
| **Ex. No.:08** | **Tree Traversal** | **Date:18/04/2024** |

**Write a C program to implement a Binary tree and perform the following tree traversal operation.**

**(i) Inorder Traversal**

**(ii) Preorder Traversal**

**(iii) Postorder Traversal**

**Algorithm:**

```
1) Start
2) Define a Node structure with data, left child pointer, and right child pointer.      3)
   Create functions for the following traversal methods:
     4) Inorder traversal:
- Recursively call the function on the left child.
- Print the data of the current Node.
- Recursively call the function on the right child.
5) Preorder traversal:
- Print the data of the current Node.
- Recursively call the function on the left child.
- Recursively call the function on the right child.
6) Postorder traversal:
- Recursively call the function on the left child.
- Recursively call the function on the right child.
- Print the data of the current Node.
7) Initialize the root of the binary tree.
8) Call the traversal functions with the root Node to perform inorder, preorder, and
   postorder traversal. 9) Stop
```

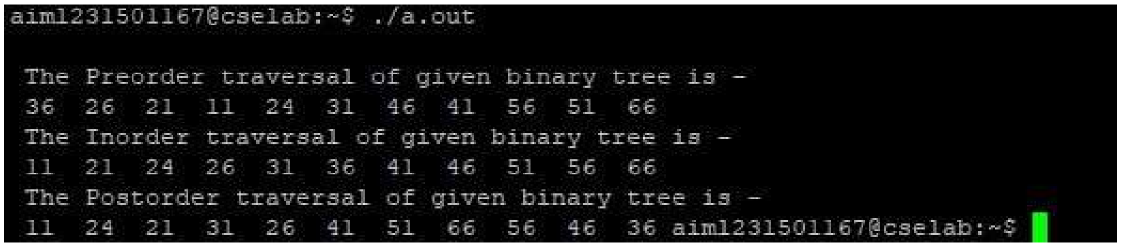
PROGRAM;

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

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

struct node* createNode(int val)
{
struct node* Node = (struct node*)malloc(sizeof(struct node));
Node->element = val;
Node->left = NULL;
Node->right = NULL;

return (Node);
}

void traversePreorder(struct node* root)
{
if (root == NULL) return;
printf(" %d ", root->element); traversePreorder(root->left);
traversePreorder(root->right);
}


void traverseInorder(struct node* root)
{
```

```
if (root == NULL) return;

traverseInorder(root->left);

printf(" %d ", root->element);

traverseInorder(root->right);

}

void traversePostorder(struct node* root)

{

if (root == NULL) return;

traversePostorder(root->left);

traversePostorder(root->right); printf("

%d ", root->element);

}


int main()

{

struct node* root = createNode(36); root-

>left = createNode(26); root->right =

createNode(46); root->left->left =

createNode(21); root->left->right =

createNode(31); root->left->left->left =

createNode(11); root->left->left->right =

createNode(24); root->right->left =

createNode(41); root->right->right =

createNode(56); root->right->right->left =

createNode(51); root->right->right->right =

createNode(66);


printf("\n The Preorder traversal of given binary tree is -\n"); traversePreorder(root);


printf("\n The Inorder traversal of given binary tree is -\n"); traverseInorder(root);


printf("\n The Postorder traversal of given binary tree is -\n");

traversePostorder(root);
```

```
return 0;
}
```

OUTPUT:



```
aiml231501167@cselab:~$ ./a.out

 The Preorder traversal of given binary tree is -
 36  26  21  11  24  31  46  41  56  51  66
 The Inorder traversal of given binary tree is -
 11  21  24  26  31  36  41  46  51  56  66
 The Postorder traversal of given binary tree is -
 11  24  21  31  26  41  51  66  56  46  36 aiml231501167@cselab:~$
```

**RESULT: Thus, the program was successfully executed.**