

Subject:
Faculty:
Topic:

Class Notes

Unit No:
Lecture No:
Link to Session
Planner (SP):S.No.... of SP
Book Reference:
Date Conducted:
Page No:

Implementation of Tree Traversal.

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

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

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

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

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

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

    return newNode;
}

// Insert on the left of the node
struct node* insertLeft(struct node* root, int value) {
    root->left = createNode(value);
```

```

return root->left;
}

// Insert on the right of the node
struct node* insertRight(struct node* root, int value) {
    root->right = createNode(value);
    return root->right;
}

int main() {
    struct node* root = createNode(1);
    insertLeft(root, 12);
    insertRight(root, 9);

    insertLeft(root->left, 5);
    insertRight(root->left, 6);

    printf("Inorder traversal \n");
    inorderTraversal(root);

    printf("\nPreorder traversal \n");
    preorderTraversal(root);

    printf("\nPostorder traversal \n");
    postorderTraversal(root);
}

```

Implementation of Binary search tree.

```

#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *left;
    struct node *right;
} *n, *temp, *root=NULL;
void create();
void inorder(struct node *temp);
void preorder(struct node *temp);
void postorder(struct node *temp);
int main()
{
    int ch;
    printf("\n1.Create\n2.Inorder\n3.Preorder\n4.Postorder\n");
    do
    {
        printf("\nEnter choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: create();
                    break;
            case 2: printf("\nInorder Traversal\n");
                    inorder(root);

```

```

        break;
case 3: printf("\nPreorder Traversal\n");
        preorder(root);
        break;
case 4: printf("\nPostorder Traversal\n");
        postorder(root);
        break;
}
} while(ch<=4);
return 0;
}
void create()
{
n=(struct node *)malloc(sizeof(struct node));
printf("\nEnter data:");
scanf("%d",&n->data);
n->left=NULL;
n->right=NULL;
if(root==NULL)
root =n;
else

{
temp=root;
while(temp!=NULL)
{
if(n->data< temp->data)
{
if(temp->left ==NULL)
{
temp->left=n;
return;
}
else
temp=temp->left;
}
else
{
if(temp->right==NULL)
{
temp->right=n;
return;
}
else
temp=temp->right;
}
}
}
}
void inorder(struct node *temp)
{
if(temp!=NULL)
{
inorder(temp->left);
printf("%d",temp->data);
inorder(temp->right);
}
}

```

```
}  
}  
void preorder(struct node *temp)  
{  
  
if(temp!=NULL)  
{  
  
printf("%d",temp->data);  
preorder(temp->left);  
preorder(temp->right);  
}  
}  
void postorder(struct node *temp)  
{  
if(temp!=NULL)  
{  
postorder(temp->left);  
postorder(temp->right);  
printf("%d",temp->data);  
}  
}
```





