Date:27.10.2022

Program: To traverse a tree using In-order, Pre-order and Post-order traversal.

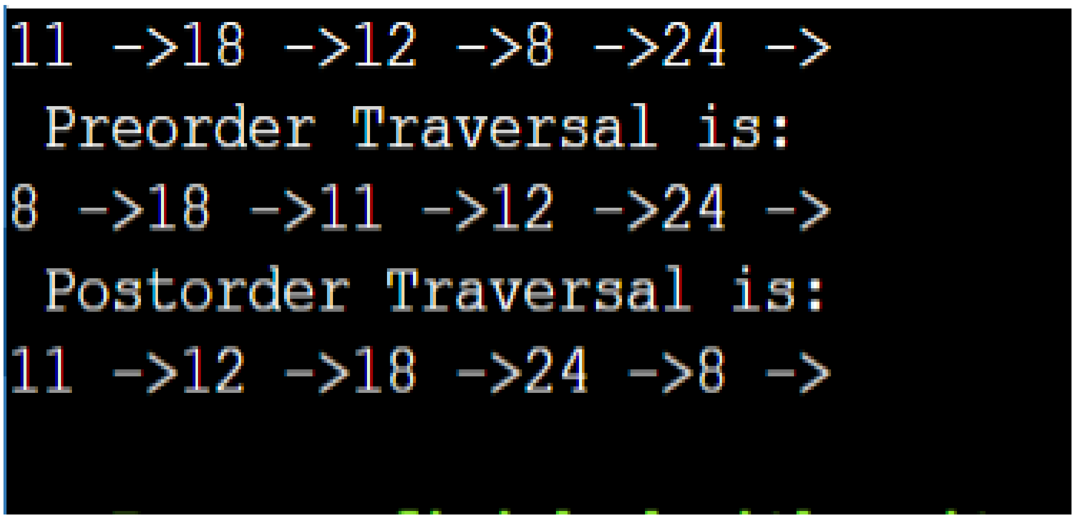
Code:

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

  int data;

  struct node* left;

  struct node* right;

};

void inorderTraversal(struct node* root)

{

  if (root == NULL)

    return;

  inorderTraversal(root->left);

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

  inorderTraversal(root->right);

}

void preorderTraversal(struct node* root)

{

  if (root == NULL)

    return;

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

  preorderTraversal(root->left);

  preorderTraversal(root->right);

}

void postorderTraversal(struct node* root)
```

```
{
   if (root == NULL)
      return;
   postorderTraversal(root->left);
   postorderTraversal(root->right);
   printf("%d ->", root->data);
}
struct node* createNode(int value)
{
   struct node* newNode = malloc(sizeof(struct node));
   newNode->data=value;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}
struct node* insertLeft(struct node* root, int value)
{
   root->left = createNode(value);
   return root->left;
}
struct node* insertRight(struct node* root, int value)
{
   root->right = createNode(value);
   return root->right;
}
int main()
{
   struct node* root = createNode(8);
   insertLeft(root, 18);
```

```
    insertRight(root, 24);

    insertLeft(root->left, 11);

    insertRight(root->left, 12);

    printf("The Inorder Traversal is:\n");

    inorderTraversal(root);

    printf("\n Preorder Traversal is:\n");

    preorderTraversal(root);

    printf("\n Postorder Traversal is:\n");

    postorderTraversal(root);

}
```

OUTPUT:



```
11 ->18 ->12 ->8 ->24 ->
 Preorder Traversal is:
8 ->18 ->11 ->12 ->24 ->
 Postorder Traversal is:
11 ->12 ->18 ->24 ->8 ->
```

Date: 03.11.2022

Program: To implement operations in Binary Search Tree (BST).

Code:

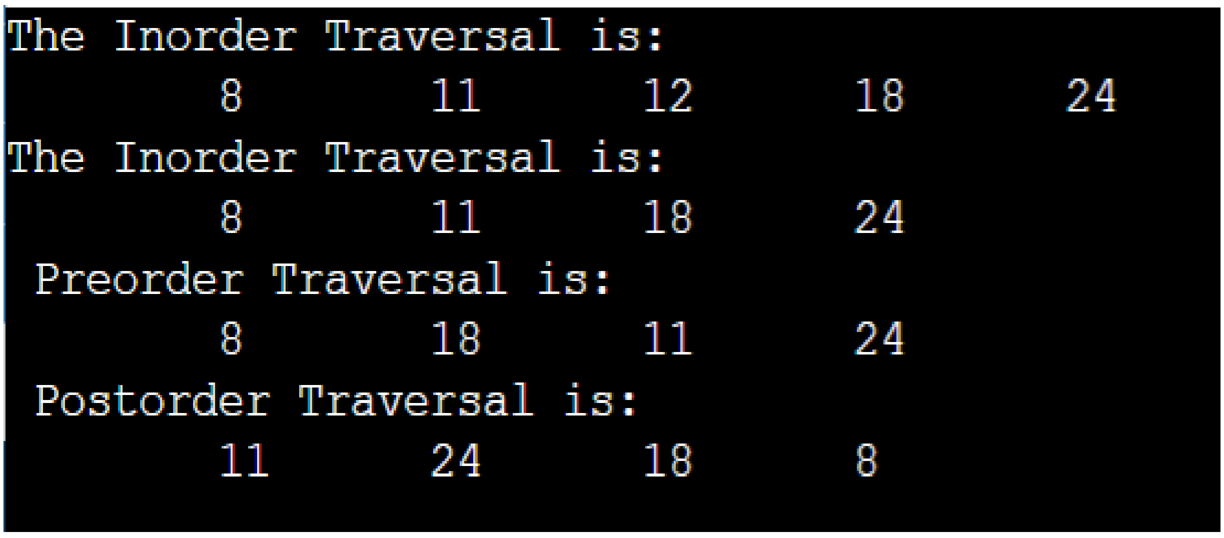
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
   struct node *left;
   int data;
   struct node *right;
};
struct node *tree=NULL;
struct node* insertelement(struct node *tree,int n)
{
   struct node *newnode,*nodeptr,*parentptr;
   newnode=(struct node *)malloc(sizeof(struct node));
   newnode->data=n;
   newnode->left=NULL;
   newnode->right=NULL;
   if(tree==NULL)
   {
      tree=newnode;
   }
   else
   {
      parentptr=NULL;
      nodeptr=tree;
      while(nodeptr!=NULL)
```

```
        {
            parentptr=nodeptr;
            if(n<nodeptr->data)
                nodeptr=nodeptr->left;
            else
                nodeptr = nodeptr->right;
        }
        if(n<parentptr->data)
            parentptr->left=newnode;
        else
            parentptr->right=newnode;
    }
    return tree;
};
struct node *minValueNode(struct node *node)
{
    struct node *current = node;
    while (current && current->left != NULL)
        current = current->left;
    return current;
}
struct node *deleteNode(struct node *root, int data)
{
    if (root == NULL)
        return tree;
    if (data < root->data)
        root->left = deleteNode(root->left, data);
    else if (data > root->data)
        root->right = deleteNode(root->right, data);
    else
    {
        if (root->left == NULL)
        {
            struct node *temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL)
        {
            struct node *temp = root->left;
            free(root);
            return temp;
        }
        struct node *temp = minValueNode(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}
int inorder(struct node *tree)
{
    while(tree!=NULL)
    {
        inorder(tree->left);
        printf("\t%d",tree->data);
        inorder(tree->right);
        return 0;
    }
```

```
}
int postorder(struct node *tree)
{
   while(tree!=NULL)
   {
      postorder(tree->left);
      postorder(tree->right);
      printf("\t%d",tree->data);
      return 0;
   }
}
int preorder(struct node *tree)
{
   while(tree!=NULL)
   {
      printf("\t%d",tree->data);
      preorder(tree->left);
      preorder(tree->right);
      return 0;
   }
}
int main()
{
   struct node* root = insertelement(root,8);
   insertelement(root, 18);
   insertelement(root, 24);
   insertelement(root, 11);
   insertelement(root, 12);
   printf("The Inorder Traversal is:\n");
```

```
    inorder(root);
    deleteNode(root,12);
    printf("\nThe Inorder Traversal is:\n");
    inorder(root);
    printf("\n Preorder Traversal is:\n");
    preorder(root);
    printf("\n Postorder Traversal is:\n");
    postorder(root);
}
```

OUTPUT:



```
The Inorder Traversal is:
        8       11      12      18      24
The Inorder Traversal is:
        8       11      18      24
 Preorder Traversal is:
        8       18      11      24
 Postorder Traversal is:
        11      24      18      8
```