Practical No : 7

Name : Sujal Sunil Nimje

Roll No : 64

Code :

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

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

typedef struct node node;

node *createnode(int data)
{
    node *root = (node *)malloc(sizeof(node));
    root->data = data;
    root->left = NULL;
    root->right = NULL;
    return root;
}

node *insert(node *root, int data)
{
    if (root == NULL)
        return createnode(data);

    else if (data > root->data)
        root->right = insert(root->right, data);

    else
        root->left = insert(root->left, data);

    return root;
}

node *creat(node *root, int size)
```

```
{
    if (size == 0) return NULL;

    int data;
    printf("enter the data of a node : ");
    scanf("%d", &data);
    root = insert(root, data);
    creat(root, size - 1);
    return root;
}

void inorder(node *root)
{
    if (root == NULL)
    {
        return;
    }

    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}
void preorder(node *root)
{
    if (root == NULL)
    {
        return;
    }

    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}
void postorder(node *root)
{
    if (root == NULL)
    {
        return;
    }

    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}

int countInternal(node *root)
{
    if (root == NULL || (root->left == NULL && root->right == NULL))
```

```
        return 0;

    return countInternal(root->left) + countInternal(root->right) + 1;
}

int countHalf(node *root)
{
    if (root == NULL || (root->left == NULL && root->right == NULL))
        return 0;

    int left = countHalf(root->left);
    int right = countHalf(root->right);

    if (root->left == NULL || root->right == NULL)
        return 1 + left + right;

    return left + right;
}

int height(node *root)
{

    if (root == NULL || (root->left == NULL && root->right == NULL))
        return 0;

    int left = height(root->left) + 1;
    int right = height(root->right) + 1;

    return left > right ? left : right;
}

void main()
{

    // node *root = NULL;
    // int myarray[] = {4, 2, 1, -1, -2};
    // for (int i = 0; i < 5; i++)
    // {
    //     root = insert(root, myarray[i]);
    // }

    // inorder(root);
    // printf("\n");
    // preorder(root);
    // printf("\n");
    // postorder(root);
    // printf("\n%d", countInternal(root) - 1);
    // printf("\n%d", countHalf(root));
```

```
    // printf("\n%d", height(root));

    node *root = NULL;

    root = creat(root, 4);
    inorder(root);
}
```

Output :

```
PS C:\Users\SUJAL NIMJE\OneDrive\Desktop\c program\BST> gcc .\bst.c
PS C:\Users\SUJAL NIMJE\OneDrive\Desktop\c program\BST> ./a.exe
enter the data of a node : 2
enter the data of a node : 1
enter the data of a node : 3
enter the data of a node : 4
1 2 3 4
2 1 3 4
1 4 3 2
 the count of internal nodes is : 1
 the count of Half nodes is : 1
 Height of a tree is : 2
PS C:\Users\SUJAL NIMJE\OneDrive\Desktop\c program\BST>
```