DS Lab Assignment 8

Pranav Joshi
CS-B Batch 2
Roll no: 43

Title: WAP to implement TBT and Perform different Traversals on it without using a Stack.

Code:

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

struct node{
    char data;
    int lbit;
    int rbit;
    struct node* left;
    struct node* right;
    struct node* parent;
};

typedef struct node node;

node* createNode(char data){
    node* newNode = (node*)malloc(sizeof(node));
    newNode->data = data;
    newNode->lbit = 0;
    newNode->rbit = 0;
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->parent = NULL;
    return newNode;
}


node* insert(node* root, char data){
    if(root == NULL){
        return createNode(data);
    }
    node* temp = root;
    node* parent = NULL;
```

```
    while(1){
        if(data < temp->data){
            if(temp->lbit == 0){
                node* newNode = createNode(data);
                newNode->left = temp->left;
                newNode->right = temp;
                newNode->parent = temp;
                temp->lbit = 1;
                temp->left = newNode;
                return root;
            }
            else{
                parent = temp;
                temp = temp->left;
            }
        }
        else{
            if(temp->rbit == 0){
                node* newNode = createNode(data);
                newNode->right = temp->right;
                newNode->left = temp;
                newNode->parent = temp;
                temp->rbit = 1;
                temp->right = newNode;
                return root;
            }
            else{
                parent = temp;
                temp = temp->right;
            }
        }
    }
}


void postorderTBT(node* root) {
    node* temp = root;
    node* lastVisited = NULL;
    while (temp->lbit == 1) {
        temp = temp->left;
    }
    while (temp != NULL) {
        if (temp->rbit == 0 || temp->right == lastVisited) {
            printf("%c ", temp->data);
            lastVisited = temp;
            temp = temp->parent;
        }
        else {
            temp = temp->right;
```

```
            while (temp->lbit == 1) {
                temp = temp->left;
            }
        }
    }
}

void inorderTBT(node* root){
    node* temp = root;
    while(temp->lbit == 1){
        temp = temp->left;
    }
    while(temp != NULL){
        printf("%c ", temp->data);
        if(temp->rbit == 0){
            temp = temp->right;
        }
        else{
            temp = temp->right;
            while(temp->lbit == 1){
                temp = temp->left;
            }
        }
    }
}

void preorderTBT(node* root){
    node* temp = root;
    while(temp != NULL){
        printf("%c ", temp->data);
        if(temp->lbit == 1){
            temp = temp->left;
        }
        else if(temp->rbit == 1){
            temp = temp->right;
        }
        else{
            while(temp != NULL && temp->rbit == 0){
                temp = temp->right;
            }
            if(temp != NULL){
                temp = temp->right;
            }
        }
    }
}

int main(){
```

```c
    node* root = NULL;
    root = insert(root, 'F');
    root = insert(root, 'C');
    root = insert(root, 'Q');
    root = insert(root, 'A');
    root = insert(root, 'D');
    root = insert(root, 'H');
    root = insert(root, 'K');

// Input Tree taken is:
//          F
//        /     \
//      C        Q
//     / \      /
//    A   D   H
//              \
//               K

    printf("Inorder traversal: ");
    inorderTBT(root);
    printf("\nPreorder traversal: ");
    preorderTBT(root);
    printf("\nPostorder traversal: ");
    postorderTBT(root);
    return 0;
}
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment8.c -o Sem4Assignment8 } ;
Inorder traversal: A C D F H K Q
Preorder traversal: F C A D Q H K
Postorder traversal: A D C K H Q F
PS C:\Code\C\Code>
```