DS Lab Assignment 6

Pranav Joshi
CS-B Batch 2
Roll no: 43

Title: WAP to create a Binary tree and perform non-recursive Preorder, Inorder and Postorder traversal on it.

Code:

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

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

struct Stack {
    int top;
    int capacity;
    struct Node **array;
};

struct Stack *createStack(int capacity) {
    struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));
    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (struct Node **)malloc(capacity * sizeof(struct Node *));
    return stack;
}

int isFull(struct Stack *stack) {
    return stack->top == stack->capacity - 1;
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}

void push(struct Stack *stack, struct Node *node) {
    if (isFull(stack))
```

```
        return;
    stack->array[++stack->top] = node;
}

struct Node *pop(struct Stack *stack) {
    if (isEmpty(stack))
        return NULL;
    return stack->array[stack->top--];
}

struct Node *createNode(int data) {
    struct Node *node = (struct Node *)malloc(sizeof(struct Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return node;
}

struct Node *insert(struct Node *root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data < root->data)
        root->left = insert(root->left, data);
    else if (data > root->data)
        root->right = insert(root->right, data);
    return root;
}

void preorder(struct Node *root) {
    if (root == NULL)
        return;
    struct Stack *stack = createStack(100);
    push(stack, root);
    while (!isEmpty(stack)) {
        struct Node *node = pop(stack);
        printf("%d ", node->data);
        if (node->right != NULL)
            push(stack, node->right);
        if (node->left != NULL)
            push(stack, node->left);
    }
}

void inorder(struct Node *root) {
    if (root == NULL)
        return;
    struct Stack *stack = createStack(100);
    struct Node *node = root;
```

```
    while (node != NULL || !isEmpty(stack)) {
        while (node != NULL) {
            push(stack, node);
            node = node->left;
        }
        node = pop(stack);
        printf("%d ", node->data);
        node = node->right;
    }
}

void postorder(struct Node *root) {
    if (root == NULL)
        return;
    struct Stack *stack1 = createStack(100);
    struct Stack *stack2 = createStack(100);
    push(stack1, root);
    while (!isEmpty(stack1)) {
        struct Node *node = pop(stack1);
        push(stack2, node);
        if (node->left != NULL)
            push(stack1, node->left);
        if (node->right != NULL)
            push(stack1, node->right);
    }
    while (!isEmpty(stack2)) {
        struct Node *node = pop(stack2);
        printf("%d ", node->data);
    }
}

int main() {
    struct Node *root = NULL;
    int n, data;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the node values:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        root = insert(root, data);
    }

// the considered tree based on user input is:
                //         60
                //       /      \
                //     40       80
                //    /  \     /   \
                //   20  50   70   90
```

```
    printf("Preorder traversal: ");
    preorder(root);

    printf("\nInorder traversal: ");
    inorder(root);

    printf("\nPostorder traversal: ");
    postorder(root);

    return 0;
}
```

Output: