#include <stdio.h>

#include <stdlib.h>

typedef struct treenode tree\_node;

struct treenode {

    int data;

    tree\_node \*left;

    tree\_node \*right;

};

tree\_node \*create\_node(int data) {

    tree\_node \*newNode = (tree\_node\*)malloc(sizeof(tree\_node));

    newNode->data = data;

    newNode->left = newNode->right = NULL;

    return newNode;

}

tree\_node\* insertLeftNode(tree\_node \*parent, int data){

    tree\_node \*left = create\_node(data);

    parent->left = left;

    return left;

}

tree\_node\* insertRightNode(tree\_node \*parent, int data){

    tree\_node \*right = create\_node(data);

    parent->right = right;

    return right;

}

void preorder\_traversal(tree\_node \*root){

    if (root != NULL) {

        printf("%d ", root->data);

        preorder\_traversal(root->left);

        preorder\_traversal(root->right);

    }

}

void inorder\_traversal(tree\_node \*root){

    if (root != NULL) {

        inorder\_traversal(root->left);

        printf("%d ", root->data);

        inorder\_traversal(root->right);

    }

}

void postorder\_traversal(tree\_node \*root){

    if (root != NULL) {

        postorder\_traversal(root->left);

        postorder\_traversal(root->right);

        printf("%d ", root->data);

    }

}

int main() {

    tree\_node \*root = create\_node(1);

    tree\_node \*left = insertLeftNode(root, 2);

    tree\_node \*right = insertRightNode(root, 3);

    insertLeftNode(left, 4);

    insertRightNode(left, 5);

    insertLeftNode(right, 6);

    insertRightNode(right, 7);

    printf("Preorder traversal: ");

    preorder\_traversal(root);

    printf("\n");

    printf("Inorder traversal: ");

    inorder\_traversal(root);

    printf("\n");

    printf("Postorder traversal: ");

    postorder\_traversal(root);

    printf("\n");

    return 0;

}

