Khwopa College of Engineering

DSA Lab Sheet

II Year/ II Part

Faculty: Computer and Electronics

Labsheet #7

Objectives:

1. Understand the structure of a Binary Tree.
2. Learn and implement traversal techniques:
    - **Inorder** (Left -> Root -> Right)
    - **Preorder** (Root -> Left -> Right)
    - **Postorder** (Left -> Right -> Root)

Theory:

A Binary Tree consists of nodes where:

- Each node contains data, left pointer, and right pointer.
- Tree traversal methods:
    - **Inorder**: Left, Root, Right
    - **Preorder**: Root, Left, Right
    - **Postorder**: Left, Right, Root

Node Structure:

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

Algorithm:

Step 1: Start.

Step 2: For INSERT operation;

If root is NULL

then create root node

return

end if

If root exists then

compare the data with node.data

while until insertion position is located

If data is greater than node.data

goto right subtree

else

goto left subtree

endwhile

insert data

end If

Step 3: For SEARCH operation;

If root.data is equal to search.data

return root

else

while data not found

If data is greater than node.data

goto right subtree

else

goto left subtree

If data found

return node

endwhile

return data not found

end if

Step 4: For IN-ORDER TRAVERSAL operation;

Until all nodes are traversed:

Recursively traverse left subtree.

Visit root node.

Recursively traverse right subtree.

Step 5: For PRE-ORDER TRAVERSAL operation;

Until all nodes are traversed:

Visit root node.

Recursively traverse left subtree.

Recursively traverse right subtree.

Step 6: For POST-ORDER TRAVERSAL operation;

Until all nodes are traversed:

Recursively traverse left subtree.

Recursively traverse right subtree.

Visit root node.

Step 7: Stop.

Execution Code:

```
#include <stdio.h>
#include <stdlib.h>
// Node structure
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
// Create new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
// Inorder traversal
void inorder(struct Node* root) {
    if (root == NULL) return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}
// Preorder traversal
void preorder(struct Node* root) {
    if (root == NULL) return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}
// Postorder traversal
void postorder(struct Node* root) {
    if (root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
```

```
// Main program
int main() {
    // Creating a sample tree
    struct Node* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);

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

    printf("\nPreorder traversal: ");
    preorder(root);

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

    printf("\n");
    return 0;
}
```

Tasks:

1. Modify the program to accept user input to build the binary tree.
2. Implement a function to count the number of nodes.
3. Implement a function to compute the height of the tree.
4. Discuss with friends how to delete a node in a binary tree.