Binary Tree Traversal :-

Ex-1 :-

Inorder Traversal :-

left → Root → Right

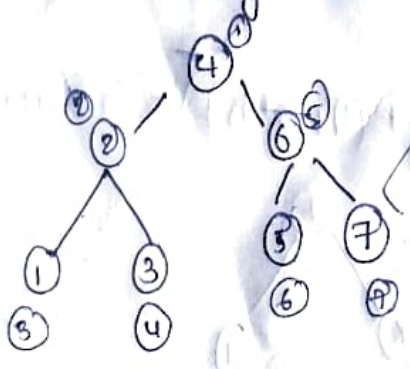
- Visit the left Subtree
- Visit the root node
- Visit the right subtree.

Inorder Traversal :- 1, 2, 3, 4, 5, 6, 7.

Preorder Traversal :-

Root → left → Right

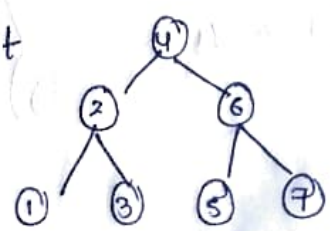
- Visit the root node.
- Visit the left subtree.
- Visit the right subtree



Preorder Traversal : 4, 2, 1, 3, 6, 5, 7

Postorder Traversal :-

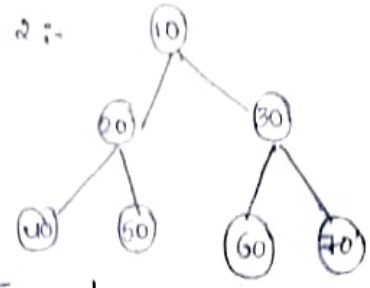
left → right → Root



- Visit the left Subtree.
- Visit the right Subtree
- Visit the root node

Post Traversal : 1, 3, 2, 5, 7, 6, 4

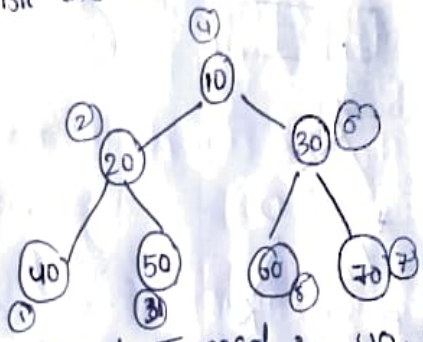
Example 2 :-



Inorder Traversal :=

left → Root → Right

- Visit the left subtree.
- Visit the right subtree.
- Visit the root node.



Inorder Traversal :- 40, 20, 50, 10, 60, 30, 70

Preorder Traversal :-

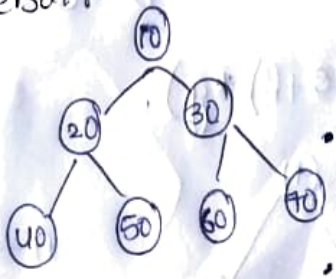
left → Root → Right

- Visit the left subtree.
- Visit the right node.
- Visit the right subtree.

Preorder Traversal :- 10, 20, 40, 50, 30, 60, 70.

Postorder Traversal :-

left → Right → Root.



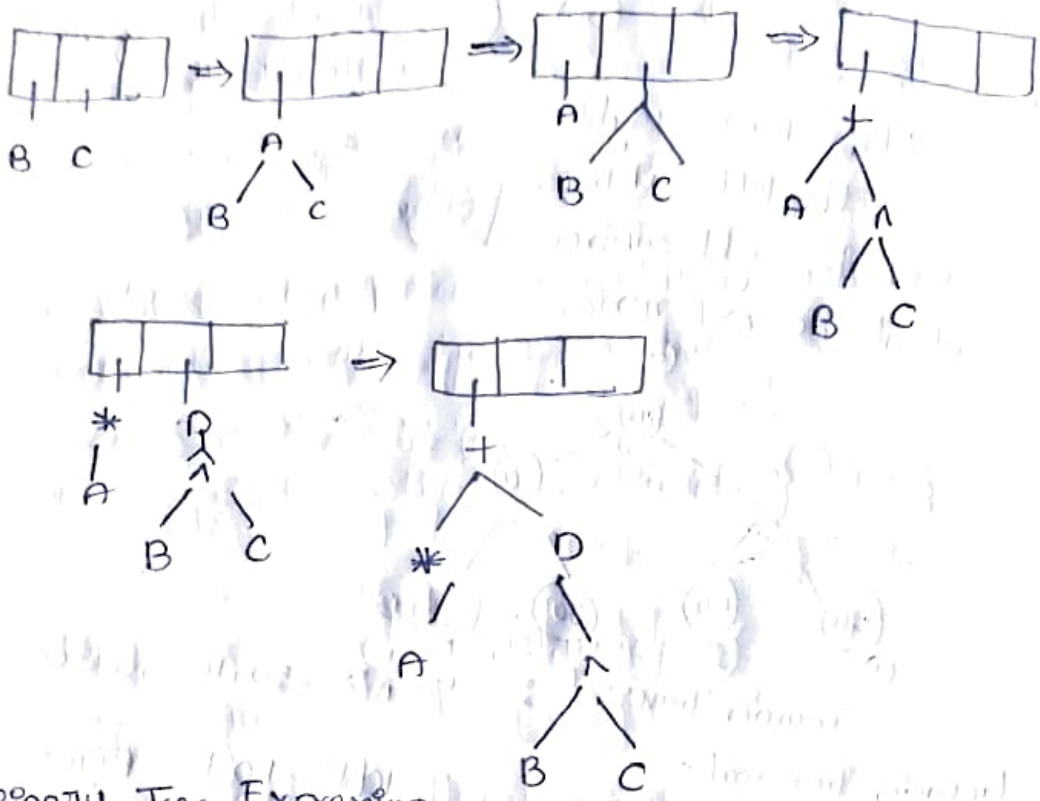
- Visit the left subtree.
- Visit the right subtree.
- Visit the root node.

Postorder Traversal : 40, 50, 20, 60, 70, 30, 10.

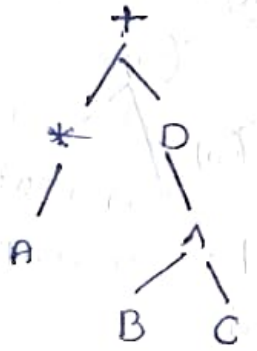
Example 3 :-

Binary Tree Expression :-

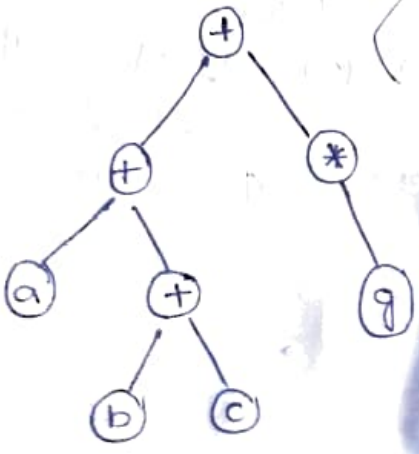
(i) A * B ^ C + D



Binary Tree Expression :=



(ii) (a+b * c) + ((d * e + f) * g).



A → R - L

* ,/ → L - R

- + → L - R

main.c

Share Run Output

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
Node* insertNode(Node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }
    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }
    return root;
```

```
/tmp/xaBuMj2EQO.o
In-order traversal of the binary tree:
20 30 40 50 60 70 80
In-order traversal after deleting 20:
30 40 50 60 70 80
In-order traversal after deleting 50:
30 40 60 70 80


=== Code Execution Successful ===
```

main.c

Share Run Output Clea

```
}
void inOrderTraversal(Node* root) {
    if (root != NULL) {
        inOrderTraversal(root->left);
        printf("%d ", root->data);
        inOrderTraversal(root->right);
    }
}
Node* findMin(Node* root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}
Node* deleteNode(Node* root, int data) {
    if (root == NULL) {
        return root;
    }
    if (data < root->data) {
        root->left = deleteNode(root->left, data);
    } else if (data > root->data) {
        root->right = deleteNode(root->right, data);
    } else {
        if (root->left == NULL) {
```

```
/tmp/xaBuMj2EQ0.o
In-order traversal of the binary tree:
20 30 40 50 60 70 80
In-order traversal after deleting 20:
30 40 50 60 70 80
In-order traversal after deleting 50:
30 40 60 70 80


=== Code Execution Successful ===
```

main.c

Share Run Output Cle

```
        if (root->left == NULL) {
            Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            Node* temp = root->left;
            free(root);
            return temp;
        }
        Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}
int main() {
    Node* root = NULL;
    root = insertNode(root, 50);
    root = insertNode(root, 30);
    root = insertNode(root, 20);
    root = insertNode(root, 40);
    root = insertNode(root, 70);
    root = insertNode(root, 60);
    root = insertNode(root, 80);
    printf("In-order traversal of the binary tree:\n");
    inOrderTraversal(root);
```

```
/tmp/xaBuMj2EQO.o
In-order traversal of the binary tree:
20 30 40 50 60 70 80
In-order traversal after deleting 20:
30 40 50 60 70 80
In-order traversal after deleting 50:
30 40 60 70 80


=== Code Execution Successful ===
```

main.c

Share Run Output

```
    }
    return root;
}
int main() {
    Node* root = NULL;
    root = insertNode(root, 50);
    root = insertNode(root, 30);
    root = insertNode(root, 20);
    root = insertNode(root, 40);
    root = insertNode(root, 70);
    root = insertNode(root, 60);
    root = insertNode(root, 80);
    printf("In-order traversal of the binary tree:\n");
    inOrderTraversal(root);
    printf("\n");
    root = deleteNode(root, 20);
    printf("In-order traversal after deleting 20:\n");
    inOrderTraversal(root);
    printf("\n");
    root = deleteNode(root, 50);
    printf("In-order traversal after deleting 50:\n");
    inOrderTraversal(root);
    printf("\n");
    return 0;
}
```

```
/tmp/xaBuMj2EQO.o
In-order traversal of the binary tree:
20 30 40 50 60 70 80
In-order traversal after deleting 20:
30 40 50 60 70 80
In-order traversal after deleting 50:
30 40 60 70 80


=== Code Execution Successful ===
```