**Experiment no: 3**

Code :-

#include <iostream>

#include <queue>

#include <stack>

#include <vector>

using namespace std;

class TreeNode {

public:

int val;

TreeNode\* left;

TreeNode\* right;

TreeNode(int v) : val(v), left(nullptr), right(nullptr) {}

};

// Function to perform BFS traversal on the tree

void bfs(TreeNode\* root) {

if (!root) return;

queue<TreeNode\*> q;

q.push(root);

cout << "BFS Traversal: ";

while (!q.empty()) {

TreeNode\* current = q.front();

q.pop();

cout << current->val << " ";

// Enqueue left and right children of the current node

if (current->left)

q.push(current->left);

if (current->right)

q.push(current->right);

}

cout << endl;

}

// Function to perform DFS traversal on the tree

void dfs(TreeNode\* root) {

if (!root) return;

stack<TreeNode\*> stk;

stk.push(root);

cout << "DFS Traversal: ";

while (!stk.empty()) {

TreeNode\* current = stk.top();

stk.pop();

cout << current->val << " ";

// Push right child first to ensure left child is processed first

if (current->right)

stk.push(current->right);

if (current->left)

stk.push(current->left);

}

cout << endl;

}

int main() {

int n;

cout << "Enter the number of nodes in the tree: ";

cin >> n;

vector<TreeNode\*> nodes(n + 1); // Vector to store pointers to nodes (1-indexed)

// Create nodes

for (int i = 1; i <= n; ++i) {

nodes[i] = new TreeNode(i);

}

// Build tree edges

cout << "Enter edges as pairs of parent and child nodes, with direction (L for left, R for right): ";

for (int i = 2; i <= n; ++i) {

int parent, child;

char direction;

cin >> parent >> child >> direction;

if (direction == 'L')

nodes[parent]->left = nodes[child];

else

nodes[parent]->right = nodes[child];

}

int choice;

do {

cout << "\nMenu:\n";

cout << "1. BFS Traversal\n";

cout << "2. DFS Traversal\n";

cout << "3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

bfs(nodes[1]); // Perform BFS traversal starting from the root (node 1)

break;

case 2:

dfs(nodes[1]); // Perform DFS traversal starting from the root (node 1)

break;

case 3:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

}

} while (choice != 3);

// Clean up memory

for (int i = 1; i <= n; ++i) {

delete nodes[i];

}

return 0;

}

**Output:-**

Enter the number of nodes in the tree: 5

Enter edges as pairs of parent and child nodes, with direction (L for left, R for right):

1 2 L

1 3 R

2 4 L

2 5 R

Menu:

1. BFS Traversal

2. DFS Traversal

3. Exit

Enter your choice: 1

BFS Traversal: 1 2 3 4 5

Menu:

1. BFS Traversal

2. DFS Traversal

3. Exit

Enter your choice: 2

DFS Traversal: 1 2 4 5 3

Menu:

1. BFS Traversal

2. DFS Traversal

3. Exit

Enter your choice: 3

Exiting...