.Q1. Create binary tree and perform recursive traversals.

/\*Create binary tree and perform recursive traversals\*/

#include <iostream>

using namespace std;

// Structure for a node of a binary tree

struct Node {

int data;

Node\* left;

Node\* right;

};

// Function to create a new node and return its address

Node\* getNewNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

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

return newNode;

}

// Recursive function to do pre-order traversal of the binary tree

void preOrder(Node\* root) {

if (root == NULL) return;

cout << root->data << " ";

preOrder(root->left);

preOrder(root->right);

}

// Recursive function to do in-order traversal of the binary tree

void inOrder(Node\* root) {

if (root == NULL) return;

inOrder(root->left);

cout << root->data << " ";

inOrder(root->right);

}

// Recursive function to do post-order traversal of the binary tree

void postOrder(Node\* root) {

if (root == NULL) return;

postOrder(root->left);

postOrder(root->right);

cout << root->data << " ";

}

int main() {

Node\* root = getNewNode(1);

root->left = getNewNode(2);

root->right = getNewNode(3);

root->left->left = getNewNode(4);

root->left->right = getNewNode(5);

cout << "Pre-order traversal: ";

preOrder(root);

cout << endl;

cout << "In-order traversal: ";

inOrder(root);

cout << endl;

cout << "Post-order traversal: ";

postOrder(root);

cout << endl;

return 0;

}

Q. 2. Write a menu driven program to perform following operations on singly linked list: Create,

reverse, search, count and Display

/\*Write a menu driven program to perform following

operations on singly linked list: Create, reverse,

search, count and Display.

\*/

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class LinkedList {

private:

Node\* head;

int count;

public:

LinkedList() {

head = NULL;

count = 0;

}

void create() {

int data;

cout << "Enter the data for the node: ";

cin >> data;

Node\* newNode = new Node();

newNode->data = data;

newNode->next = head;

head = newNode;

count++;

}

void reverse() {

Node\* prev = NULL;

Node\* current = head;

Node\* next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

int search(int key) {

Node\* current = head;

int index = 0;

while (current != NULL) {

if (current->data == key) {

return index;

}

current = current->next;

index++;

}

return -1;

}

int countNodes() {

return count;

}

void display() {

Node\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

};

int main() {

int choice;

LinkedList list;

while (true) {

cout << "1. Create Node" << endl;

cout << "2. Reverse List" << endl;

cout << "3. Search Element" << endl;

cout << "4. Count Nodes" << endl;

cout << "5. Display List" << endl;

cout << "6. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

list.create();

break;

case 2:

list.reverse();

break;

case 3: {

int key;

cout << "Enter the element to be searched: ";

cin >> key;

int index = list.search(key);

if (index == -1) {

cout << "Element not found." << endl;

} else {

cout << "Element found at index: " << index << endl;

}

break;

}

case 4:

cout << "Number of nodes: " << list.countNodes() << endl;

break;

case 5:

list.display();

break;

case 6:

return 0;

default:

cout << "Invalid choice. Please enter a valid choice." << endl;

}

}

return 0;

}