Q. 1. 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. Implement Linked queue

/\*. Implement Linked queue\*/

#include <bits/stdc++.h>

using namespace std;

struct QNode {

int data;

QNode\* next;

QNode(int d)

{

data = d;

next = NULL;

}

};

struct Queue {

QNode \*front, \*rear;

Queue() { front = rear = NULL; }

void enQueue(int x)

{

// Create a new LL node

QNode\* temp = new QNode(x);

// If queue is empty, then

// new node is front and rear both

if (rear == NULL) {

front = rear = temp;

return;

}

// Add the new node at

// the end of queue and change rear

rear->next = temp;

rear = temp;

}

// Function to remove

// a key from given queue q

void deQueue()

{

// If queue is empty, return NULL.

if (front == NULL)

return;

// Store previous front and

// move front one node ahead

QNode\* temp = front;

front = front->next;

// If front becomes NULL, then

// change rear also as NULL

if (front == NULL)

rear = NULL;

delete (temp);

}

};

// Driver code

int main()

{

Queue q;

q.enQueue(10);

q.enQueue(20);

q.deQueue();

q.deQueue();

q.enQueue(30);

q.enQueue(40);

q.enQueue(50);

q.deQueue();

cout << "Queue Front : " << (q.front)->data << endl;

cout << "Queue Rear : " << (q.rear)->data;

}