**SUBMITTED BY : HARSHA VARDHAN**

QUES.1

#Code:

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node \*left, \*right;

Node(int data)

{

this->data = data;

left = right = NULL;

}

};

// Preorder traversal

void preorderTraversal(struct Node \*node)

{

if (node == NULL)

return;

cout << node->data << "->";

preorderTraversal(node->left);

preorderTraversal(node->right);

}

// Postorder traversal

void postorderTraversal(struct Node \*node)

{

if (node == NULL)

return;

postorderTraversal(node->left);

postorderTraversal(node->right);

cout << node->data << "->";

}

// Inorder traversal

void inorderTraversal(struct Node \*node)

{

if (node == NULL)

return;

inorderTraversal(node->left);

cout << node->data << "->";

inorderTraversal(node->right);

}

int main()

{

struct Node \*root = new Node(1);

root->left = new Node(12);

root->right = new Node(9);

root->left->left = new Node(5);

root->left->right = new Node(6);

cout << "Inorder traversal ";

inorderTraversal(root);

cout << "\nPreorder traversal ";

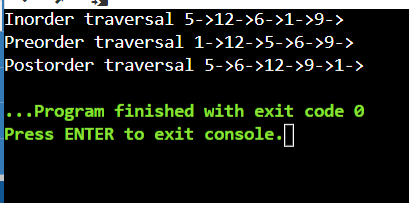
preorderTraversal(root);

cout << "\nPostorder traversal ";

postorderTraversal(root);

}

OUTPUT:



Ques.2

#code:

#include<iostream>

using namespace std;

struct node

{

int key;

struct node \*left, \*right;

};

// Create a node

struct node \*newNode(int item)

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

// Inorder Traversal

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

cout << root->key << " -> ";

inorder(root->right);

}

}

// Insert a node

struct node \*insert(struct node \*node, int key)

{

if (node == NULL)

return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

return node;

}

struct node \*minValueNode(struct node \*node)

{

struct node \*current = node;

// Find the leftmost leaf

while (current && current->left != NULL)

current = current->left;

return current;

}

// Deleting a node

struct node \*deleteNode(struct node \*root, int key)

{

if (root == NULL)

return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else

{

if (root->left == NULL)

{

struct node \*temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL)

{

struct node \*temp = root->left;

free(root);

return temp;

}

struct node \*temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

int main()

{

struct node \*root = NULL;

root = insert(root, 5);

root = insert(root, 4);

root = insert(root, 5);

root = insert(root, 6);

root = insert(root, 10);

root = insert(root, 1);

root = insert(root, 24);

root = insert(root, 5);

cout << "Inorder traversal: ";

inorder(root);

cout << "\nAfter deleting 10\n";

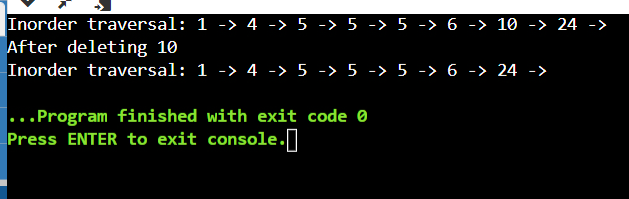
root = deleteNode(root, 10);

cout << "Inorder traversal: ";

inorder(root);

}

OUTPUT:



Ques.2

#code:

#include<bits/stdc++.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

Node(int d) {

data = d;

next = NULL; }};

void insert(Node \*&head, int val){

Node \*n = new Node(val);

if (head == NULL) {

head = n;

return;

}

Node \*temp = head;

while (temp->next != NULL) {

temp = temp->next; }

temp->next = n;}

Node \*merge(Node \*&head1, Node \*&head2){

Node \*p1 = head1;

Node \*p2 = head2;

Node \*dummyNode = new Node(-1);

Node \*p3 = dummyNode;

while (p1 != NULL && p2 != NULL)

{

if (p1->data < p2->data)

{

p3->next = p1;

p1 = p1->next;

}

else

{

p3->next = p2;

p2 = p2->next; }

p3 = p3->next; }

while (p1 != NULL)

{

p3->next = p1;

p1 = p1->next;

p3 = p3->next; }

while (p2 != NULL) {

p3->next = p2;

p2 = p2->next;

p3 = p3->next;

}

return dummyNode->next;

}

void display(Node \*head)

{

Node \*temp = head;

while (temp != NULL)

{

cout << temp->data << "-> ";

temp = temp->next;

}

cout << "NULL" << endl;

}

int main(){

Node \*head1 = NULL;

Node \*head2 = NULL;

int arr1[] = {1, 5, 8, 9};

int arr2[] = {2, 3, 6};

for (int i = 0; i < 4; i++) {

insert(head1, arr1[i]); }

display(head1);

for (int i = 0; i < 3; i++) {

insert(head2, arr2[i]); }

display(head2);

Node \*newhead = merge(head1, head2);

display(newhead);

}

OUTPUT:

