Assignment #4

Question #3:

#include<iostream>

using namespace std;

class Node

{

public:

int data;

Node\* left;

Node\* right;

Node(int d)

{

data = d;

left = NULL;

right = NULL;

}

};

void preorder(Node\* root)

{

if (root == NULL)

{

return;

}

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

preorder(root->left);

preorder(root->right);

}

void inorder(Node\* root)

{

if (root==NULL)

{

return;

}

inorder(root->left);

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

inorder(root->right);

}

void postorder(Node\* root)

{

if (root==NULL)

{

return;

}

postorder(root->left);

postorder(root->right);

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

}

int main()

{

Node\* root = new Node(1);

root->left = new Node(2);

root->right = new Node(3);

root->left->left = new Node(4);

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

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

root->right->right = new Node(7);

root->right->left->left = new Node(8);

root->right->left->right = new Node(9);

int choice;

cout << "Enter the Choice " << endl;

cout << "1. To Check preorder Nodes " << endl;

cout << "2. To Check Inorder Nodes " << endl;

cout << "3. To Check Post Order Nodes " << endl;

cin >> choice;

if (choice==1)

{

cout << "The Nodes of preoder : ";

preorder(root);

cout << endl;

}

else if (choice == 2)

{

cout << "The Nodes of Inorder : ";

inorder(root);

cout << endl;

}

else if (choice == 3)

{

cout << "The Nodes of Postorder : ";

postorder(root);

cout << endl;

}

else

{

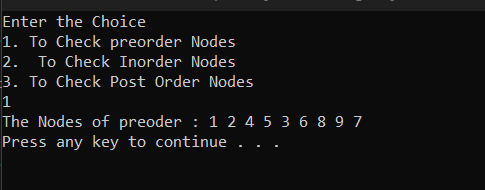
cout << "Sorry Wrong Input " << endl;

}

system("pause");

}

Output:



Assignment #4:

#include<iostream>

#include<algorithm>

using namespace std;

struct Node

{

public:

int data;

Node\*left, \*right;

Node(int va)

{

data = va;

left = NULL;

right = NULL;

}

};

Node\* insertBSt(Node\* root, int val)

{

if (root == NULL)

{

return new Node(val);

}

else if (val < root->data)

{

root->left = insertBSt(root->left, val);

}

else

{

root->right = insertBSt(root->right, val);

}

return root;

}

//void inorder(Node\* root)

//{

// if (root == NULL)

// {

// return;

// }

// inorder(root->left);

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

// inorder(root->right);

//}

Node\* searchinBST(Node\* root,int key)

{

if (root==NULL)

{

return NULL;

}

if (root->data==key)

{

return root;

}

//key>data

if (root->data>key)

{

return searchinBST(root->left, key);

}

//data<key

return searchinBST(root->right, key);

}

Node\* inordersucc(Node\* root)

{

Node\* curr = root;

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

{

curr = curr->left;

}

return curr;

}

Node\* deleteinBST(Node\* root, int key)

{

if (key<root->data)

{

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

}

else if (key > root->data)

{

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

}

else

{

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;

}

//case 3

Node\* temp = inordersucc(root->right);

root->data = temp->data;

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

}

return root;

}

int countNodes(Node\* root)

{

if (root==NULL)

{

return 0;

}

return countNodes(root->left) + countNodes(root->right) + 1;

}

void inorder(Node\* root)

{

if (root==NULL)

{

return;

}

inorder(root->left);

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

inorder(root->right);

}

int sumNodes(Node\* root)

{

if (root==NULL)

{

return 0;

}

return sumNodes(root->left) + sumNodes(root->right) + root->data;

}

int caluheight(Node\* root)

{

if (root==NULL)

{

return 0;

}

int lHeight = caluheight(root->left);

int rHeight = caluheight(root->right);

return max(lHeight, rHeight) + 1;

}

int main()

{

//Node\* root = new Node(4);

/\*root->left = new Node(2);

root->right = new Node(5);

root->left->left = new Node(1);

root->left->right = new Node(3);

root->right->right = new Node(6);\*/

Node\* root = NULL;

root = insertBSt(root, 3);

insertBSt(root,2);

insertBSt(root, 5);

insertBSt(root,4);

insertBSt(root, 6);

cout << "The inorder traversal is " << endl;

inorder(root);

cout << endl;

cout << "The number of Nodes in the tree : "<<countNodes(root)<< " " << endl;

cout << "The height of the tree is : " << caluheight(root) << " " << endl;

cout << endl;

if (searchinBST(root,5)==NULL)

{

cout << "Key does not exsist " << endl;

}

else

{

cout << "key exsist " << endl;

}

root = deleteinBST(root, 5);

//inorder(root);

cout << "The sum of Nodes = " << sumNodes(root) << endl; //b part

system("pause");

}

Output:

