**Data Structures and Algorithms**

**Lab-10**

Ahmad Amjad Mughal

121672

BSCS-6C

Task:

#include <iostream> //Input/output Library

using namespace std; //for prompt statements

struct node \*tree = NULL; //tree is empty means tree pointer is pointing nothing

struct node //node predefiniton

{

int data;

node \*left;

node \*right;

};

struct node\* newNode(int data) //node creation function that uses data as input argument

{

struct node\* tree = new(struct node);

tree->data = data;

tree->left = NULL;

tree->right = NULL;

return(tree);

}

//for inserting a new node

struct node\* insert(struct node\* tree, int data)

{

if (tree == NULL) //if tree is pointing NULL

{

return(newNode(data)); //new node is created

}

else

{

if (data <= tree->data) //if data value of new node is less than that of tree node

tree->left = insert(tree->left, data); //lef part holds the adress of that new value

else

tree->right = insert(tree->right, data); //else data value of new node is greater than that of tree node

return(tree);

}

}

void Inorder(struct node\* tree)

{

if (tree == NULL) //for root tree

return;

Inorder(tree->left); //computing Traversal in Inorder

cout << tree->data;

Inorder(tree->right);

}

//for printing Postorder Traversal

void Postorder(struct node\* tree)

{

if (tree == NULL)

return;

Postorder(tree->left); //computing Traversal in Postorder

Postorder(tree->right);

cout << tree->data;

}

//for printing Preorder Traversal

void Preorder(struct node\* tree)

{

if (tree == NULL)

return;

cout << tree->data; //computing Traversal in Preorder

Postorder(tree->left);

Postorder(tree->right);

}

bool search(node \*tree, int data)

{

if (tree == NULL)

return false;

else if (data == tree->data)

cout<<”searched”<,endl;

return true;

else if (data <= tree->data)

return search(tree->left, data);

else

return search(tree->right, data);

}

void printTree(struct node\*tree)

{

if (tree == NULL) return;

printTree(tree->left);

cout << tree->data;

printTree(tree->right);

}

int main()

{

cout << "The Binary tree nodes are " << endl;

node\* tree = newNode(5);

insert(tree, 7), insert(tree, 2), insert(tree, 8), insert(tree, 3),

insert(tree, 6);

printTree(tree);

cout << endl;

cout << "Preorder Traversal is: ";

Preorder(tree);

cout << endl << "InOrder Traversal is: ";

Inorder(tree);

cout << endl << "PostOrder Traversal is:";

Postorder(tree);

cout << endl;

search(tree,5);

}

Screenshot:

