**Name: Abdul Ghaffar Kalhoro**

**Registration # 194699**

**Class: BSCS-6C**

Data structures and Algorithms

lab 9 Binary Search Tree operations

**Source Code:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Abdul Ghaffar Kalhoro \*

reg # 194699 \*

BSCS-6C \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

lab 9 Binary Search Tree operations \*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tree Operations \*

1)Insertion \*

2)Search \*

3)Pre Order display \*

4)In Order display \*

5)Post order display \*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include<iostream>

using namespace std;

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

// Function to Insert Node in a Binary Search Tree

Node\* Insert(Node \*root, int value) {

if (root == NULL) {

root = new Node();

root->data = value;

root->left = root->right = NULL;

}

else if (value <= root->data)

root->left = Insert(root->left, value);

else

root->right = Insert(root->right, value);

return root;

}

// Preorder display function

void Preorder\_func(struct Node \*root) {

// base condition

if (root == NULL) return;

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

Preorder\_func(root->left); // use root->left as a parameter

Preorder\_func(root->right); // use root->right as a parameter

}

//Inorder display function

void Inorder\_func(Node \*root) {

//base condition

if (root == NULL) return;

Inorder\_func(root->left); // use root->left as a parameter of inorder function

cout << root->data << ", "; //print the data

Inorder\_func(root->right); // use root->right as a parameter of Inorder function

}

//Postorder display function

void Postorder\_func(Node \*root) {

if (root == NULL) return;

Postorder\_func(root->left); // use root->left as a parameter of postorder

Postorder\_func(root->right); // use root->right as a parameter of postorder

cout << root->data << ", "; // Print the data

}

//searching node value function

bool searchNode(Node \*root, int data)

{

if (root == NULL)

return false;

else if (data == root->data)

return true;

else if (data <= root->data)

return searchNode(root->left, data);

else

return searchNode(root->right, data);

}

void displaychoice(){

cout << "What do you want to do: " << endl;

cout << "1) Insert new node" << endl;

cout << "2) display in Pre\_Order form" << endl;

cout << "3) display in In\_Order form" << endl;

cout << "4) display in Post\_Order form" << endl;

cout << "5) search from the tree" << endl;

cout << "6) exit";

cout << "\n";

}

int main() {

Node\* root = NULL;

int element;

int numbers;

int inputdata;

int choice;

cout << "~~~BINARY SEARCH TREE~~~"<<endl;

cout << "Enter numbers to make initial tree: ";

cin >> numbers;

cout << "Enter data elements: ";

for (int counter=1;counter <=numbers;counter++){

cin >> inputdata;

root = Insert(root, inputdata);

}

displaychoice();

cin >> choice;

while (choice != 6){

switch (choice){

case 1:

cout << "Enter value to insert in a tree: ";

cin >> inputdata;

root = Insert(root, inputdata);

cout << "\n"; cout << "\n"; cout << "\n";

break;

case 2:

cout << endl << "Preorder: ";

Preorder\_func(root);//calling the preorder function

cout << "\n"; cout << "\n"; cout << "\n";

break;

case 3:

cout << "Inorder: ";

Inorder\_func(root); //using inorder function

cout << "\n"; cout << "\n"; cout << "\n";

break;

case 4:

cout << "Postorder: ";

Postorder\_func(root); //postorder function calling

cout << "\n"; cout << "\n"; cout << "\n";

break;

case 5:

cout << "Enter any element to search from the tree: ";

cin >> element;

if (searchNode(root, element)){

cout << "The entered value = " << element << " is present in our tree"<<endl<<endl;

}

else

cout << "The entered value = " << element << " does not exist"<<endl<<endl;

default:

cout << "wrong choice entered.!!"<<endl<<endl;

break;

}

displaychoice();

cin >> choice;

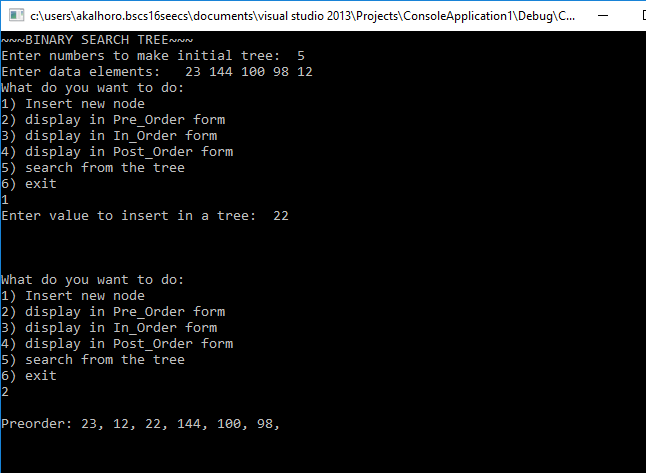
}

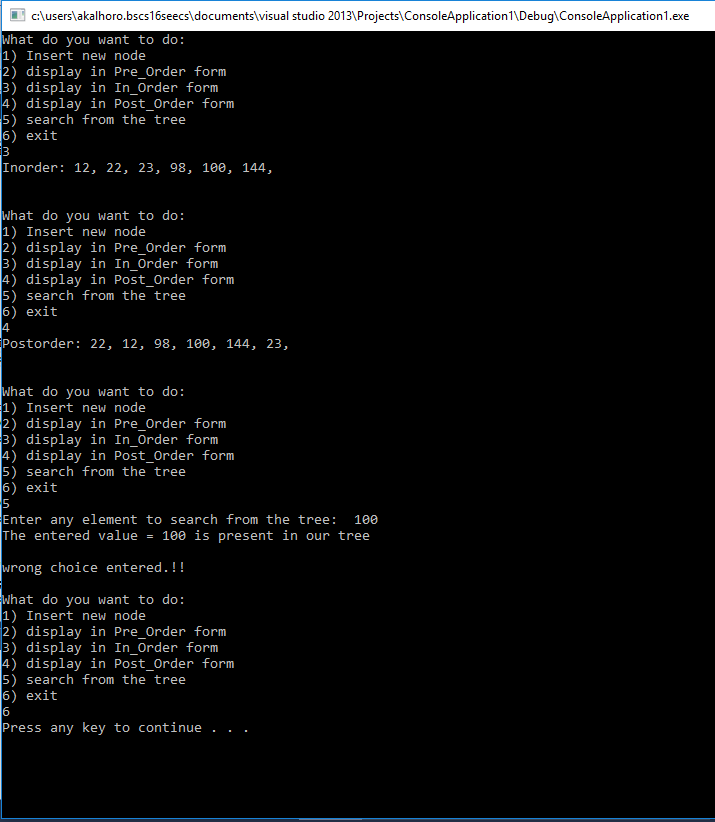
system("pause");

return 0;

}

**OUTPUT**

****

****