**Bahria University, Lahore Campus**

Department of Computer Science

Lab Journal 11

**(Spring 2023)**

|  |  |  |
| --- | --- | --- |
| Course: | **Data Structures and Algorithm - Lab** | Date: \_5-25-20223\_\_ |
| Course Code: | CSL-221 | Max Marks: 10 |
| Faculty’s Name: | Fatima Zulfiqar |  |

Name: Affan ahmad \_\_ Enroll No: \_03-134221-003\_\_ Class: \_BS(cs\_\_\_

Objective(s):

Upon completion of this lab session, learners will be able to:

* Implement Binary Search Tree (BST)
* Insert given node in a BST
* Delete nodes from BST
* Find maximum and minimum number from the BST

## Lab Tasks:

**Task 1**

Write a program to implement Binary Search Tree using following given data elements. Also display elements in the tree using inorder traversal.

**45 15 79 90 10 55 12 20 50 90 79 6**

Task no1and 2.

#include <iostream>

using namespace std;

struct node {

int data;

node \*left;

node \*right;

};

node \*create(int val)

{

node \* newnode = new node;

newnode->data = val;

newnode->right = NULL;

newnode->left = NULL;

return newnode;

}

node \*insert( node \*node, int data) {

if (node == NULL)

{

return create(data);

}

if (data < node->data)

{

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

}

else

{

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

}

return node;

}

void inorder(node\* root)

{

if (root != NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

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

struct node \*current = node;

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

current = current->left;

return current;

}

struct node \*deleteNode(struct node \*root, int val) {

if (root == NULL) return root;

if (val < root->data)

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

else if (val > root->data)

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

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->data = temp->data;

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

}

return root;

}

int main()

{

int v,val;

node \*root = NULL;

root = insert(root, 45);

root = insert(root, 15);

root = insert(root, 79);

root = insert(root, 90);

root = insert(root, 10);

root = insert(root, 55);

root = insert(root, 12);

root = insert(root, 20);

root = insert(root, 50);

root = insert(root, 90);

root = insert(root, 79);

root = insert(root, 6);

cout << "press 1 for insertion :" << endl;

cout << "press 2 for traversal :" << endl;

cout << "press 3 for delete node :" << endl;

cout << "press 4 for insertion :" << endl;

cout << "press 5 for exit :" << endl;

do

{

cout << "enter your choise :"; cin >> v;

if (v == 1)

{

cout << "enter your value :"; cin >> val;

root = insert(root, val);

cout << endl;

}

if (v == 2)

{

inorder(root);

}

if (v == 3)

{

root=deleteNode(root,20);

inorder(root);

}

if (v == 5)

{

cout << "exit"<< endl;

break;

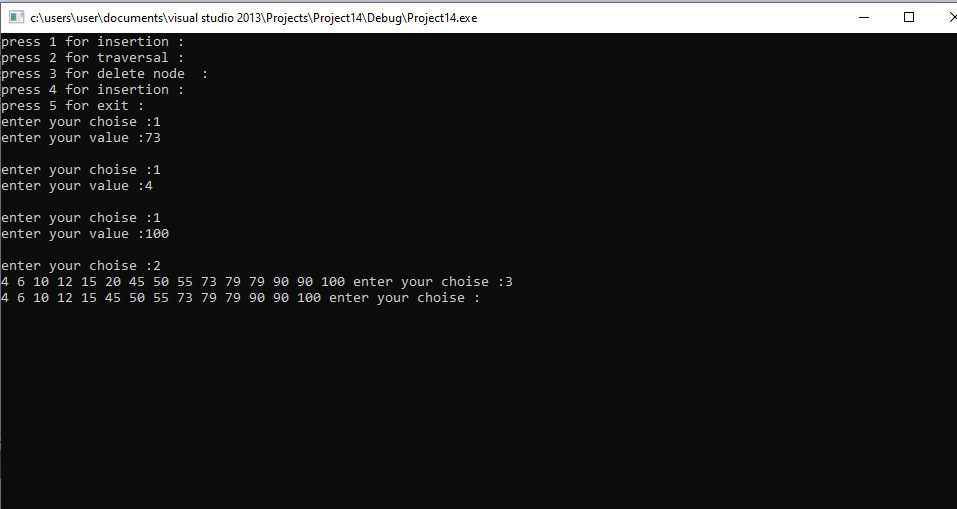
}

} while (v != 5);

system("pause");

return 0;

}



**Task 2**

Using a tree obtained in **TASK 1**, Implement a function to insert new node in the given BST. Lets suppose, new elements to be inserted is **73, 4,** and **100.** Also display resultant tree.

**Task 3**

Implement a function to delete nodes from the given BST. You are required to delete node with value **20** from the BST obtained in **TASK** **2**. Also display the resultant tree.

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

struct node \*current = node;

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

current = current->left;

return current;

}

struct node \*deleteNode(struct node \*root, int val) {

if (root == NULL) return root;

if (val < root->data)

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

else if (val > root->data)

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

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->data = temp->data;

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

}

return root;

}

**Task 4**

Write an algorithm to find maximum and minimum number from the tree obtained after completing **TASK 3**.

int findMax(node\* root)

{

if (root == NULL)

return 0;

int res = root->data;

int lres = findMax(root->left);

int rres = findMax(root->right);

if (lres > res)

res = lres;

if (rres > res)

res = rres;

return res;

}

int findMin(node \*root)

{

if(root==NULL)

{

return INT\_MAX;;

}

int res=root->data;

int left=findMin(root->left);

int right=findMin(root->right);

if(left<res)

{

res=left;

}

if(right<res)

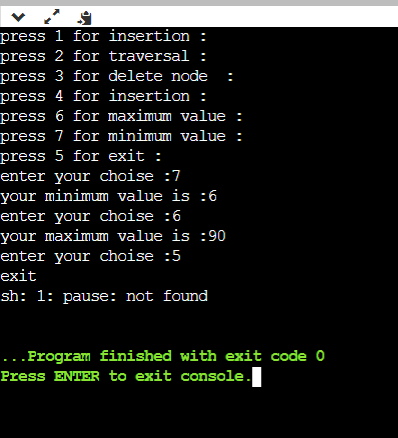
{

res=right;

}

return res;

}



**Note : Attempt all tasks and get them checked by your Lab Instructor. Also for each task, attach a screenshot of the output.**

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 2 |  |  |
| 2. | 2 |  |  |
| 3. | 2 |  |  |
| 4. | 4 |  |  |
| **Total** | **10** |  | **Signature** |