**Bahria University, Lahore Campus**

Department of Computer Science

Lab Journal 10

**(Spring 2023)**

|  |  |  |
| --- | --- | --- |
| Course: | **Data Structures and Algorithm - Lab** | Date: \_21-05-2023\_\_\_ |
| Course Code: | CSL-221 | Max Marks: 10 |
| Faculty’s Name: | Fatima Zulfiqar |  |

Name: AFFAN AHMAD \_\_\_ Enroll No: \_03-134221-003\_\_\_\_ Class: \_BS(cs )3A\_\_

Objective(s):

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

* Count number of leaf nodes in the binary tree
* Find height of the given tree
* Find the level of specific node in a binary tree
* Find nodes present in any specific level of a binary tree
* Find maximum depth of a binary tree

## Lab Tasks:

**Task 1**

Write a program to implement a given binary tree data structure



**Task 2**

Write a function to count and return total number of leaf nodes in a binary tree obtained in **Task 1. And task 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;

}

void preorder(node\* root)

{

if (root != NULL)

{

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

preorder(root->left);

preorder(root->right);

}

}

void inorder(node\* root)

{

if (root != NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

void postorder(node\* root)

{

if (root != NULL)

{

postorder(root->left);

postorder(root->right);

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

}

}

void searching( node \*root, int val )

{

if (root != NULL)

{

if (root->data == val)

{

cout << val << " is found " << endl;

}

searching(root->left, val);

searching(root->right, val);

}

else

{

cout << "elsement not found " << endl;

}

}

int getLeafCount(struct node\* node)

{

if(node == NULL)

return 0;

if(node->left == NULL && node->right == NULL)

return 1;

else

return getLeafCount(node->left)+

getLeafCount(node->right);

}

int main()

{

int v,val;

node \*newnode = create (1);

//root left

newnode->left = create(2);

newnode->left->right = create(5);

newnode->left->right->left = create(10);

newnode->left->right->right = create(11);

newnode->left->left = create(4);

newnode->left->left->right = create(9);

newnode->left->left->left = create(8);

// root right

newnode->right = create(3);

newnode->right->left = create(6);

newnode->right->right = create(7);

newnode->right->left->left = create(13);

newnode->right->right->right=create(14);

cout << "pree 1 for pre order " << endl;

cout << "pree 2 for in order " << endl;

cout << "pree 3 for post order " << endl;

cout << "pree 5 for searching using preorder " << endl;

cout << "pree 6 for check leaf nodes " << endl;

cout << "pree 4 for exit " << endl;

do

{

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

if (v == 1)

{

cout << "pre order traversal is :" << endl;

preorder(newnode);

cout << endl;

}

if (v == 2)

{

cout << "in order traversal is :" << endl;

inorder(newnode);

cout << endl;

}

if (v == 3)

{

cout << "post order traversal is :" << endl;

postorder(newnode);

cout << endl;

}

if (v == 5)

{

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

searching(newnode, val);

}

if (v == 6)

{

cout << "total leafnodes is ";

cout << getLeafCount(newnode)<< endl;

}

if (v == 4)

{

cout << "EXIT " << endl;

break;

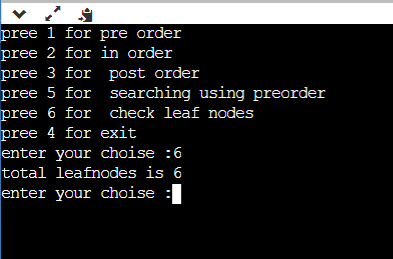
}

} while (v != 4);

system("pause");

return 0;

}



**Task 3**

Write an algorithm to find the maximum depth of a binary tree. Also write a formula to find maximum depth of a binary tree.

int findDepth(node\* root, int x)

{

if (root == NULL)

return -1;

int dist = -1;

if ((root->data == x)

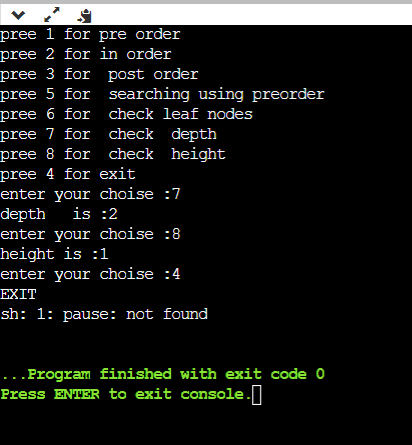
|| (dist = findDepth(root->left, x)) >= 0

|| (dist = findDepth(root->right, x)) >= 0)

return dist + 1;

return dist;

}



**Task 4**

Write an algorithm to find the height of a binary tree given in **Task 1**. Also write a formula to calculate height of a binary tree.

int findHeightUtil(node\* root, int x,

int& height)

{

if (root == NULL) {

return -1;

}

int leftHeight = findHeightUtil(

root->left, x, height);

int rightHeight

= findHeightUtil(

root->right, x, height);

int ans = max(leftHeight, rightHeight) + 1;

if (root->data == x)

height = ans;

return ans;

}

int findHeight(node\* root, int x)

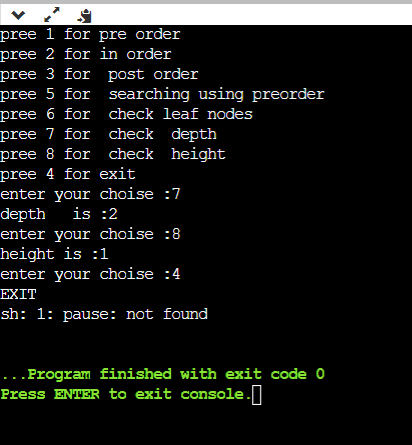
{

int h = -1;

int maxHeight = findHeightUtil(root, x, h);

return h;

}



**Task 5**

Write an algorithm to find level of specific element present in a binary tree.

int getLevelUtil(struct node\* node, int data, int level)

{

if (node == NULL)

return 0;

if (node->data == data)

return level;

int downlevel

= getLevelUtil(node->left, data, level + 1);

if (downlevel != 0)

return downlevel;

downlevel = getLevelUtil(node->right, data, level + 1);

return downlevel;

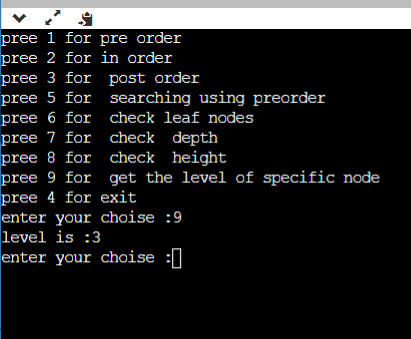
}

int getLevel(struct node\* node, int data)

{

return getLevelUtil(node, data, 1);

}



**Task 6**

Write an algorithm to display all the nodes present in any given level of a binary tree.

**void printNodesAtLevel(struct node\* root, int currentLevel, int level) {**

**if(root == NULL) {**

**return;**

**}**

**if(currentLevel == level)**

**{**

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

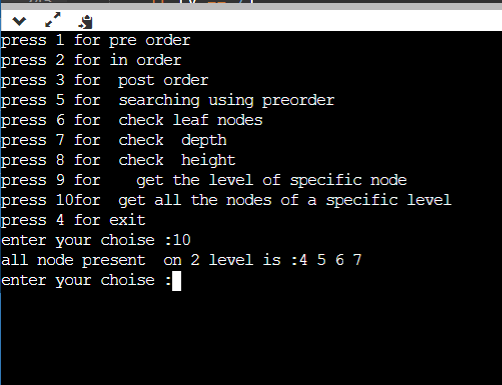
**return;**

**}**

**printNodesAtLevel(root->left, currentLevel+1, level);**

**printNodesAtLevel(root->right, currentLevel+1, level);**

**}**



**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 0 |  |  |
| 2. | 2 |  |  |
| 3. | 2 |  |  |
| 4. | 2 |  |  |
| 5. | 2 |  |  |
| 6. | 2 |  |  |
| **Total** | **10** |  | **Signature** |

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