Data Structures and Algorithms

Lab 11: Implementation of Binary Trees

**TASKS**

**Task #.1: Estimated Time: 30 Mins.**

Write C++ code to implement binary trees in C++ using Arrays.

Code:

#include<iostream>

using namespace std;

char tree[10];

int

root (char key)

{

if (tree[0] != '\0')

cout << "Tree already had root";

else

tree[0] = key;

return 0;

}

int

set\_left (char key, int parent)

{

if (tree[parent] == '\0')

cout << "\nCan't set child at:( "

<<(parent \* 2) + 1

<<" , no parent found:(";

else

tree[(parent \* 2) + 1] = key;

return 0;

}

int

set\_right (char key, int parent)

{

if (tree[parent] == '\0')

cout << "\nCan't set child at "

<<(parent \* 2) + 2

<<" , no parent found";

else

tree[(parent \* 2) + 2] = key;

return 0;

}

int

print\_tree ()

{

cout << "\n";

for (int i = 0; i < 10; i++)

{

if (tree[i] != '\0')

cout << tree[i];

else

cout << "-";

}

return 0;

}

int

main ()

{

root ('A');

set\_right ('C', 0);

set\_left ('D', 1);

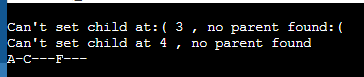
set\_right ('E', 1);

set\_right ('F', 2);

print\_tree ();

return 0;

}



**Task #.2: Estimated Time: 30 Mins.**

Write C++ code to implement binary trees in C++ using Linked List.

**Code**

#include <iostream>

#include <string>

#include <queue>

using namespace std;

struct ListNode

{

int data;

ListNode \* next;

};

struct BinaryTreeNode

{

int data;

BinaryTreeNode \* left, \*right;

};

void

push (struct ListNode \*\*head\_ref, int new\_data)

{

struct ListNode \*new\_node = new ListNode;

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

BinaryTreeNode \* newBinaryTreeNode (int data)

{

BinaryTreeNode \* temp = new BinaryTreeNode;

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

void

convertList2Binary (ListNode \* head, BinaryTreeNode \* &root)

{

queue < BinaryTreeNode \* >q;

if (head == NULL)

{

root = NULL;

return;

}

root = newBinaryTreeNode (head->data);

q.push (root);

head = head->next;

while (head)

{

BinaryTreeNode \* parent = q.front ();

q.pop ();

BinaryTreeNode \* leftChild = NULL, \*rightChild = NULL;

leftChild = newBinaryTreeNode (head->data);

q.push (leftChild);

head = head->next;

if (head)

{

rightChild = newBinaryTreeNode (head->data);

q.push (rightChild);

head = head->next;

}

parent->left = leftChild;

parent->right = rightChild;

q.pop ();

}

}

void

inorderTraversal (BinaryTreeNode \* root)

{

if (root)

{

inorderTraversal (root->left);

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

inorderTraversal (root->right);

}

}

int

main ()

{

struct ListNode \*head = NULL;

push (&head, 36);

push (&head, 30);

push (&head, 15);

push (&head, 16);

push (&head, 12);

push (&head, 10);

BinaryTreeNode \* root;

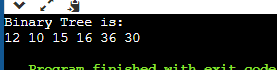
convertList2Binary (head, root);

cout << "Binary Tree is: \n";

inorderTraversal (root);

return 0;

}



**Task #.3: Estimated Time: 30 Mins.**

Write C++ code to implement binary trees using

1. Inorder Traversal
2. Preorder Traversal
3. Postorder Traversal

**Code:**

#include <iostream>

#include <string>

#include <queue>

using namespace std;

struct ListNode

{

int data;

ListNode\* next;

};

struct BinaryTreeNode

{

int data;

BinaryTreeNode \*left, \*right;

};

void push(struct ListNode\*\* head\_ref, int new\_data)

{

struct ListNode\* new\_node = new ListNode;

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

BinaryTreeNode\* newBinaryTreeNode(int data)

{

BinaryTreeNode \*temp = new BinaryTreeNode;

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

void convertList2Binary(ListNode \*head, BinaryTreeNode\* &root)

{

queue<BinaryTreeNode \*> q;

if (head == NULL)

{

root = NULL;

return;

}

root = newBinaryTreeNode(head->data);

q.push(root);

head = head->next;

while (head)

{

BinaryTreeNode\* parent = q.front();

q.pop();

BinaryTreeNode \*leftChild = NULL, \*rightChild = NULL;

leftChild = newBinaryTreeNode(head->data);

q.push(leftChild);

head = head->next;

if (head)

{

rightChild = newBinaryTreeNode(head->data);

q.push(rightChild);

head = head->next;

}

parent->left = leftChild;

parent->right = rightChild;

q.pop();

}

}

void inorderTraversal(BinaryTreeNode\* root)

{

if (root)

{

inorderTraversal( root->left );

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

inorderTraversal( root->right );

}

}

int main()

{

struct ListNode\* head = NULL;

push(&head, 36);

push(&head, 30);

push(&head, 25);

push(&head, 15);

push(&head, 12);

push(&head, 10);

BinaryTreeNode \*root;

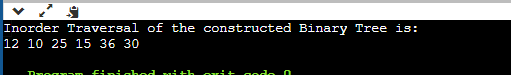
convertList2Binary(head, root);

cout << "Inorder Traversal of the constructed Binary Tree is: \n";

inorderTraversal(root);

return 0;

}



Department of Computer Sciences, BULC **1**