**Assignment number: 2**

**Subject: ADVANCED DATA STRUCTURES LAB**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**PROBLEM STATEMENT:**

Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

**Code:**

#include<iostream>

using namespace std;

class TBT;

class node

{

node \*left, \*right;

bool rbit, lbit;

int data;

friend class TBT;

};

class TBT

{

node \*root, \*dummy;

node \*\*arr;

public:

TBT()

{

root = NULL;

dummy = new node;

dummy->left = NULL;

dummy->right = NULL;

dummy->data = -9999;

}

int n;

int z=0;

node \*New, \*temp;

void create(int n);

void inorder\_array\_push(node\* );

node \*set\_root();

void thread(int n);

bool check\_suc(node\* ,node\* );

bool check\_pre(node\* ,node\* );

void inorder\_TBT();

void arr\_display();

node \*findmin(node\* );

};

void TBT :: arr\_display()

{

cout<<arr[0]->data;

cout<<arr[1]->data;

cout<<arr[2]->data;

cout<<arr[3]->data;

cout<<arr[4]->data<<endl;

}

node\* TBT :: findmin(node\* root)

{

while(root->lbit == true){

root=root->left;

}

return root;

}

void TBT :: inorder\_TBT()

{

node \*t = new node;

t = arr[0];

while(t!=dummy){

if(t->rbit == true){

cout<<t->data<<" ";

t = findmin(t->right);

}

cout<<t->data<<" ";

t = t->right;

}

cout<<endl;

}

node\* TBT :: set\_root()

{

return root;

}

void TBT :: create(int n)

{

if(root)

{

cout<<"tree is already"<<endl;

return;

}

else{

int value;

arr = new node\*[n];

while(n){

cout<<"Enter a num: ";

cin>>value;

New = new node;

New->data = value;

New->left = NULL;

New->right = NULL;

New->lbit = true;

New->rbit = true;

if(root == NULL)

root = New;

else{

temp = root;

while(1){

if(temp->data < value){

if(temp->right == NULL){

temp->right = New;

break;

}

else{

temp = temp->right;

}

}

else{

if(temp->left == NULL){

temp->left = New;

break;

}

else{

temp = temp->left;

}

}

}

}

n--;

}

}

}

void TBT :: inorder\_array\_push(node \*Root){

if(Root){

inorder\_array\_push(Root->left);

arr[z] = Root;

z++;

inorder\_array\_push(Root->right);

}

}

void TBT :: thread(int n){

int x=0;

while(x<n){

if(arr[x]->left && arr[x]->right){

cout<<":)left thread: "<<arr[x]->lbit<<"right thread: "<<arr[x]->rbit<<endl;

cout<<":) x++"<<endl;

x++;

}

else{

if(x == 0){

arr[x]->left = dummy;

arr[x]->lbit = false;

cout<<":) x=0 & arr[]->left == NULL"<<endl;

}

if(arr[x]->left == NULL && x!=0){

arr[x]->left = arr[x-1];

arr[x]->lbit = false;

cout<<":) arr[]->left == NULL"<<endl;

}

if(arr[x]->right == NULL){

arr[x]->right = arr[x+1];

arr[x]->rbit = false;

cout<<":) arr[]->right == NULL"<<endl;

}

if(x == n-1){

arr[x]->right = dummy;

arr[x]->rbit = false;

cout<<":) x=n-1"<<endl;

break;

}

cout<<":) x++"<<endl;

x++;

}

}

}

bool TBT :: check\_suc(node \*a, node\* b){

if(a->right == NULL)

return false;

else{

if(a->right == b || a->left == b || a==b)

return true;

if(a->right->data > b->data)

a=a->right;

check\_suc(a->left, b);

}

}

bool TBT :: check\_pre(node \*a, node\* b){

if(a->left == NULL)

return false;

else{

if(a->right == b || a->left == b || a==b)

return true;

if(a->right->data > b->data)

a=a->right;

check\_suc(a->left, b);

}

}

int main(){

TBT T;

int n;

cout<<"Enter number of Nodes: ";

cin>>n;

T.create(n);

T.inorder\_array\_push(T.set\_root());

T.thread(n);

T.inorder\_TBT();

return 0;

}

**OUTPUT:**

Enter number of Nodes: 6

Enter a num: 12

Enter a num: 78

Enter a num: 55

Enter a num: 33

Enter a num: 28

Enter a num: 14

:) x=0 & arr[]->left == NULL

:) x++

:) arr[]->left == NULL

:) arr[]->right == NULL

:) x++

:) arr[]->right == NULL

:) x++

:) arr[]->right == NULL

:) x++

:) arr[]->right == NULL

:) x++

:) arr[]->right == NULL

:) x=n-1

12 14 28 33 55 78