* **TBT.h**

/\*

\* TBT.h

\* Created on: Nov 25, 2020

\* Author: Megha Sonavane

\*/

**#ifndef** TBT\_H\_

**#define** TBT\_H\_

**struct** Node{

**int** data;

Node\* left;

Node\*right;

**bool** lThread;

**bool** rThread;

};

**class** TBT {

**public**:

Node\*root;

**TBT**();

Node\* **getNode**(**int** key);

**bool** **insert**(**int** key);

**void** **inorder**();

Node\* **inOrderSuccessor**(Node\*);

**void** **preorder**();

**virtual** **~TBT**();

};

**#endif** /\* TBT\_H\_ \*/

* **TBT.cpp**

/\*

\* TBT.cpp

\* Created on: Nov 25, 2020

\* Author: Megha Sonavane

\*/

**#include**<iostream>

**#include** "TBT.h"

**using** **namespace** std;

**TBT::TBT**() {

// **TODO** Auto-generated constructor stub

root=NULL;

}

//=================definition of getNode()================================

Node\* **TBT::getNode**(**int** key){

Node\* T=**new** Node;

T->data=key;

T->lThread=**true**;

T->rThread=**true**;

**return** T;

}

//===============definition of insert()===================================

**bool** **TBT::insert**(**int** key){

//if tree is not empty

Node\*ptr=root;

**while**(ptr!=NULL){

**if**(key==ptr->data)

{

**return** **false**;

}

**if**(key < ptr->data)

{

**if**(ptr->lThread==**false**)

ptr=ptr->left;

**else**

**break**;

}

**else**{

**if**(ptr->rThread==**false**)

ptr=ptr->right;

**else**

**break**;

}

}

Node\* newN=getNode(key);

**if**(ptr==NULL){

root=newN;

newN->left=NULL;

newN->right=NULL;

}

**else** **if**(key<ptr->data){

newN->left=ptr->left;

newN->right=ptr;

ptr->lThread=**false**;

ptr->left=newN;

}

**else**{

newN->left=ptr;

newN->right=ptr->right;

ptr->rThread=**false**;

ptr->right=newN;

}

**return** **true**;

}

//==============definition of inorder succeesor()========================

Node\* **TBT::inOrderSuccessor**(Node\* n){

//if node has rThread then its right element is its inorder successor

**if**(n->rThread)

**return** n->right;

//else find the leftmost element from node's right subtree

n=n->right;

**while**(n->lThread==**false**)

n=n->left;

**return** n;

}

//=============inorder traversal================================

**void** **TBT::inorder**(){

**if**(root==NULL)

{

cout<<"\tTree is empty"<<**endl**;

**return**;

}

**else**{

Node\* curr=root;

//find the first element i.e leftmost element

**while**(curr->lThread==**false**)

curr=curr->left;

**while**(curr!=NULL)

{

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

//find its inorder successor

curr=inOrderSuccessor(curr);

}

}

}

//==========prorder traversal=========================================

**void** **TBT::preorder**(){

**if**(root==NULL)

{

cout<<"\tEmpty tree"<<**endl**;

**return**;

}

**else**{

Node\* curr=root;

**while**(curr!=NULL)

{

//print root data

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

//if is has left child i.e.lThread is 0, move to left

**if**(curr->lThread==**false**)

curr=curr->left;

//else move to right subtree

**else**{

//first move to root and then to right part

**while**((curr->rThread==**true**)&& (curr->right!=NULL))

{

curr=curr->right;

}

**if**(curr!=NULL)

curr=curr->right;

}

}

}

}

**TBT::~TBT**() {

// **TODO** Auto-generated destructor stub}

* **Assignment6.cpp**

//============================================================================

// Name : Assignment6.cpp

// Author : Megha Sonavane

// Description : Threaded Binary Tree

//============================================================================

**#include** <iostream>

**#include** "TBT.h"

**using** **namespace** std;

**int** **main**() {

TBT tbt;

**int** ch,n;

**bool** flag;

**do**{

cout<<**endl**<<"=============================================================================="<<**endl**;

cout<<"\t1:Insert into tree"<<**endl**<<"\t2:Inorder traversal of tree"<<**endl**<<"\t3:Preorder traversal"<<**endl**<<"\t0:Exit"<<**endl**;

cout<<"\tEnter choice:";

cin>>ch;

cout<<"=============================================================================="<<**endl**;

**switch**(ch){

**case** 1:

//======insertion in tree=========

cout<<"\tEnter number:";

cin>>n;

flag=tbt.insert(n);

**if**(flag)

cout<<"\t\*\*\*Inserted successfully\*\*\*"<<**endl**;

**else**

cout<<"\t\*\*\*"<<n<<" is alreay present in tree\*\*\*"<<**endl**;

**break**;

**case** 2:

//========inorder traversal======

tbt.inorder();

**break**;

**case** 3:

tbt.preorder();

**break**;

}

}**while**(ch!=0);

**return** 0;

}

* **Output:**

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:20

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:10

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:30

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:5

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:16

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:2

==============================================================================

5 10 16 20 30

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:3

==============================================================================

20 10 5 16 30

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:1

==============================================================================

Enter number:37

\*\*\*Inserted successfully\*\*\*

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:2

==============================================================================

5 10 16 20 30 37

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:3

==============================================================================

20 10 5 16 30 37

==============================================================================

1:Insert into tree

2:Inorder traversal of tree

3:Preorder traversal

0:Exit

Enter choice:0

==============================================================================