/\*

\* PH 43 Aditya Singh

\* Implementation of threaded binary tree and it's traversal

\*/

#include <iostream>

#include <cstring>

using namespace std;

class node {

int val;

node \*left, \*right;

bool lthread, rthread, postvisit;

friend class tree;

public:

node();

};

// init each node with threads as true

node :: node() {

lthread = rthread = true;

postvisit = false;

}

class tree {

node \*head;

public:

tree();

void create();

void preorder();

void inorder();

node\* insuccess(node\*);

void postorder();

void leafly();

};

// init bst, head points to head at first

tree :: tree() {

head = new node();

head->rthread = false; // head's right is false even if it points to itself

head->left = head->right = head;

head->postvisit = true;

}

// creates a new node in BST

void tree :: create() {

node \*curr;

curr = new node();

cout<<"\nEnter data for root ?"<<endl;

cin>>curr->val;

head->lthread = false; // since root is connected to left of head

curr->left = curr->right = head;

head->left = curr;

char ch;

do {

int flag = 0;

node \*temp;

temp = new node();

cout<<"\nEnter data for current node ?"<<endl;

cin>>temp->val;

while(flag == 0) { // if flag is set to 1 that means node is inserted so exit from loop

char choice = 'l';

cout<<"Would you like to traverse left or right ?"<<endl;

cin>>choice;

if(tolower(choice) == 'l') {

if(curr->lthread == true) {

temp->right = curr; // right points to inorder successor so its parent

temp->left = curr->left; // left points to inorder predecessor so left of curr or thread to head

curr->left = temp; // left of current now points to new node

curr->lthread = false;

flag = 1;

} else {

curr = curr->left;

}

} else {

if(curr->rthread == true) {

temp->left = curr; // inorder predecessor on right side is the parent

temp->right = curr->right; // on right inorder successor is the head itself or the right of curr

curr->right = temp;

curr->rthread = false;

flag = 1;

} else {

curr = curr->right;

}

}

}

cout<<"Would you like to continue ?"<<endl;

cin>>ch;

} while(tolower(ch) == 'y');

}

// preorder traversal

void tree :: preorder() {

node \*temp = head->left;

while(temp != head) {

cout<<temp->val<<" "; // print the value at current

while(temp->lthread == false) { // while left subtree exists

temp = temp->left;

cout<<temp->val<<" "; // print leftmost val

}

while(temp->rthread == true) {

temp = temp->right; // if right is threaded then use it to go to previous node

}

temp = temp->right; // move to right after going leftmost

}

}

// inorder traversal

void tree :: inorder() {

node \*temp = head;

while(1) {

temp = insuccess(temp);

if(temp == head) break; // if we reach head we have traversed everything

cout<<temp->val<<" ";

}

}

// determines next node while traversing inorder

node\* tree :: insuccess(node \*a) {

node \*curr = a->right; // right of start is head itself otherwise right node

if(a->rthread == false) { // if right isnt threaded

while(curr->lthread == false) {

curr = curr->left; // we move to left till we reach leaf

}

}

return curr;

}

/\*

\* The below function is just an attempt, it does not work for large tbts

\* postorder traversal

\*/

void tree :: postorder() {

node \*temp = head->left;

int prev = 0;

while(temp != head) {

while(temp->lthread == false) {

temp = temp->left; // we move to left most

}

cout<<temp->val<<" "; // print its value

if(prev != 0) {

cout<<prev<<" "; // print the value at prev which is right value

prev = 0;

}

while(temp->rthread == true) { // if rthread exists we goto inorder successor

temp = temp->right;

}

prev = temp->val; // we store this value in prev

temp = temp->right; // we move to right sub tree

}

}

// prints the leaf nodes of a tree

void tree :: leafly() {

node \*temp = head;

while(1) {

temp = insuccess(temp);

if(temp == head) break;

if(temp->lthread == true && temp->rthread == true) // if its both pointers are threaded then it is at leaf

cout<<temp->val<<" ";

}

}

int main() {

cout<<"This program performs various operations on threaded binary search tree."<<endl;

int choice = 0;

tree b;

do {

cout<<"\n\nPlease enter your choice ?\n1. Create a binary tree\n2. Perform preorder traversal\n3. Perform inorder traversal\n4. Perform postorder traversal\n5. Print leaves\n6. Exit\n"<<endl;

cin>>choice;

switch (choice) {

case 1:

b.create();

break;

case 2:

b.preorder();

break;

case 3:

b.inorder();

break;

case 4:

b.postorder();

break;

case 5:

b.leafly();

break;

case 6:

cout<<"GOODBYE !!!"<<endl;

choice = 0;

break;

default:

cout<<"\nWrong Choice !!! Please try again"<<endl;

break;

}

} while (choice != 0);

return 0;

}

/\* OUTPUT

This program performs various operations on threaded binary search tree.

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

1

Enter data for root ?

16

Enter data for current node ?

17

Would you like to traverse left or right ?

l

Would you like to continue ?

y

Enter data for current node ?

19

Would you like to traverse left or righ\_t ?

r

Would you like to continue ?

y

Enter data for current node ?

20

Would you like to traverse left or right ?

l

Would you like to traverse left or right ?

l

Would you like to continue ?

n

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

2

16 17 20 19

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

3

20 17 16 19

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

4

20 17 19 16

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

5

20 19

Please enter your choice ?

1. Create a binary tree

2. Perform preorder traversal

3. Perform inorder traversal

4. Perform postorder traversal

5. Print leaves

6. Exit

6

GOODBYE !!!

\*/