/\* Consider threading a binary tree using pre-order threads rather than inorder threads,design an algorithm for traversal without

using stack and analyze its complexity

\*/

#include<iostream>

using namespace std;

class node

{

public:

int data;

int lbit,rbit;

node \*left,\*right;

};

class TBT

{

node \*root,\*dummy;

public:

void create(int num);

void insert(node \*trav,node \*temp);

void preorder();

void inorder(node \*trav,node \*dummy);

void postorder(node \*trav,node \*dummy);

void display();

TBT()

{

root=NULL;

dummy=NULL;

}

}\*root,\*dummy;

void TBT::create(int num)

{

node \*trav,\*temp;

temp=new node();

temp->data=num;

temp->lbit=0;

temp->rbit=0;

if(root==NULL)

{

dummy=new node;

dummy->data=-9999;

dummy->lbit=1;

dummy->rbit=1;

dummy->left=temp;

dummy->right=dummy;

temp->left=dummy;

temp->right=dummy;

root=temp;

}

else

{

trav=root;

insert(trav,temp);

}

}

void TBT::insert(node \*trav,node \*temp)

{

if(temp->data<trav->data)

{

if(trav->lbit==0)

{

temp->left=trav->left;

temp->right=trav;

trav->left=temp;

trav->lbit=1;

}

else

{

insert(trav->left,temp);

}

}

if(temp->data>trav->data)

{

if(trav->rbit==0)

{

temp->right=trav->right;

temp->left=trav;

trav->right=temp;

trav->rbit=1;

}

else

{

insert(trav->right,temp);

}

}

}

void TBT::preorder()

{

node \*trav;

trav=root;

while(trav!=dummy)

{

cout<<trav->data<<endl;

if(trav->lbit==1)

{

trav=trav->left;

}

else

{

while(trav->rbit==0 && trav->right!=dummy)

{

trav=trav->right;

}

trav=trav->right;

}

}

}

void TBT::inorder(node \*trav,node \*dummy)

{

while(trav!=dummy)

{

while(trav->lbit==1)

{

trav=trav->left;

}

cout<<trav->data<<endl;

while(trav!=dummy)

{

if(trav->rbit==1)

{

trav=trav->right;

while(trav->lbit==1)

{

trav=trav->left;

}

cout<<trav->data<<endl;

}

else

{

while(trav->rbit==0)

{

trav=trav->right;

if(trav==dummy)

{

break;

}

cout<<trav->data<<endl;

}

}

}

}

}

void TBT::postorder(node \*trav,node \*dummy)

{

if(root==NULL)

cout<<"Empty Tree";

else

{

int a[20],i=0;

while(trav!=dummy)

{

a[i]=trav->data;

i++;

if(trav->rbit==1)

{

trav=trav->right;

}

else

{

while(trav->lbit==0 && trav->left!=dummy)

{

trav=trav->left;

}

trav=trav->left;

}

}

int n=i-1;

for(i=n;i>=0;i--)

{

cout<<a[i]<<endl;

}

}

}

void TBT::display()

{

int n,i,data;

cout<<"\n Enter the number nodes \n";

cin>>n;

cout<<"\n Enter the data in the nodes \n";

for(i=0;i<n;i++)

{

cin>>data;

create(data);

}

cout<<"\n Preorder: \n";

preorder();

cout<<"\n Inorder: \n";

inorder(root,dummy);

cout<<"\n Postorder: \n";

postorder(root,dummy);

}

int main()

{

TBT T;

T.display();

return 0;

}

/\*OUTPUT

Enter the number nodes

5

Enter the data in the nodes

5

6

10

23

2

Preorder:

5

2

6

10

23

Inorder:

2

5

6

10

23

Postorder:

2

23

10

6

5

Process returned 0 (0x0) execution time : 11.558 s

Press any key to continue.\*/