NAME- Aniket Kalbhor

PRN- 12210601

ROLL- CSB-48

CODE- **6.Write a menudriven program in C to perform following operations on TBT**

**a.Create b. Insert c.Inorder traversal d.Preprder traversl e.Postorder traversal h.Exit**

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

struct TBT{

int data;

bool lbit;

bool rbit;

struct TBT\* l;

struct TBT\* r;

};

struct TBT\* head;

void preOrder(struct TBT \*head){

struct TBT \*temp;

temp = head->l;

while(temp != head){

printf("%d ", temp->data);

if(temp->lbit == 1)

temp = temp->l;

else if(temp->rbit == 1)

temp = temp->r;

else{

while(temp->rbit == 0)

temp = temp->r;

temp = temp->r;

}

}

}

struct TBT\* getNode(int data);

struct TBT\* insert(struct TBT\* head , int key);

struct TBT\* inorderPredecessor(struct TBT\* p);

struct TBT\* inorderSuccessor(struct TBT\*p);

void Traversal(struct TBT\* head);

void inorder(struct TBT\* head);

struct TBT\* findParent(struct TBT\* p);

struct TBT\* postSuccessor(struct TBT\* p);

void postorder(struct TBT\* head);

int main(){

head = (struct TBT\*)malloc(sizeof(struct TBT));

head->lbit= 0;

head->rbit= 1;

head->r=head;

head->l=head;

while(1){

printf("\n-----------------------------------\n");

printf("Select operation");

printf("\n1.Create Tree/Insert node\n2.Traverse tree\n0.Exit\n\nEnter your choice: ");

int n;

scanf("%d",&n);

int temp;

switch(n){

case 1:

{

printf("\nInsert -1 to stop. \n");

while(temp!=-1)

{

printf("Enter number to Insert: ");

scanf("%d",&temp);

if(temp!=-1)

head = insert(head,temp);

}

}

break;

case 2:

Traversal(head);

break;

case 0:

exit(0);

break;

default:

printf("indataid Choixe.");

break;

}

}

return 0;

}

struct TBT\* getNode(int data){

struct TBT\* temp= (struct TBT\*)malloc(sizeof(struct TBT));

temp->data = data;

temp->lbit = 0;

temp->rbit = 0;

temp->l = NULL;

temp->r= NULL;

return temp;

}

struct TBT\* insert(struct TBT\* head , int key){

struct TBT\* temp = getNode(key);

struct TBT\* p;

if(head->l == head) {

head->l = temp;

temp->l = head;

temp->r = head;

return head;

}

p= head->l;

while(1){

if(key < p->data && p->lbit==1)

p=p->l;

else if(key > p->data && p->rbit==1)

p=p->r;

else

break;

}

if(key < p->data )

{

p->lbit = 1;

temp->l = p->l;

temp->r = p;

p->l = temp;

}

else if( key > p->data )

{

p->rbit = 1;

temp->r = p->r;

temp->l = p;

p->r = temp;

}

return head;

}

struct TBT\* inorderPredecessor(struct TBT\* p){

if(p->lbit==0)return p->l;

else if(p->lbit==1){

p=p->l;

while(p->rbit==1)p=p->r;

}

return p;

}

struct TBT\* inorderSuccessor(struct TBT\*p){

if(p->rbit==0)return p->r;

else if (p->rbit==1)

{

p=p->r;

while(p->lbit==1)p=p->l;

}

return p;

}

void inorder(struct TBT\* head){

struct TBT\* p;

p=head->l;

while(p->lbit==1)p=p->l;

while(p!=head){

printf(" %d", p->data);

p=inorderSuccessor(p);

}

}

struct TBT\* findParent(struct TBT\* p){

struct TBT\* child= p;

//ancestor of child

while(p->rbit==1)p=p->r;

p=p->r;

if(p->l==child)return p;

p=p->l;

while(p->r != child){

p=p->r;

}

return p;

}

struct TBT\* postSuccessor(struct TBT\* p){

struct TBT\* cur= p;

struct TBT\* parent = findParent(cur);

if(parent->r==cur)return parent;

else {

while(p->rbit==1)p=p->r;

p=p->r;

if(p->rbit==1){

p=p->r;

while(!(p->rbit==0 && p->lbit==0)){

if(p->lbit==1)p=p->l;

else if(p->rbit==1)p=p->r;

}

}

}

return p;

}

void postorder(struct TBT\* head)

{

struct TBT\* p = head->l;

struct TBT\* temp=p;

while(!(p->rbit==0 && p->lbit==0)){

if(p->lbit==1){

p=p->l;

}

else if(p->rbit==1)p=p->r;

}

printf(" %d", p->data);

while(p!=head->l){

p=postSuccessor(p);

printf(" %d", p->data);

}

}

void Traversal(struct TBT\* head){

printf("\nTraversal Type : \n1.Preorder\n2.Inorder\n3.PostOrder\n\n\nEnter your choice: ");

int n;

scanf("%d",&n);

switch (n)

{

case 1:

printf("\nPreorder:\n\t");

preorder(head);

preOrder(head);

break;

case 2:

printf("\nInorder:\n\t");

inorder(head);

inOrder(head);

break;

case 3:

printf("\nPostorder:\n\t");

postorder(head);

break;

default:

break;

}

}

**OUTPUT:**

-----------------------------------

Select operation

1.Create Tree/Insert node

2.Traverse tree

0.Exit

Enter your choice: 1

Insert -1 to stop.

Enter number to Insert: 60

Enter number to Insert: 40

Enter number to Insert: 69

Enter number to Insert: 65

Enter number to Insert: 70

Enter number to Insert: 77

Enter number to Insert: -1

-----------------------------------

Select operation

1.Create Tree/Insert node

2.Traverse tree

0.Exit

Enter your choice: 2

Traversal Type :

1.Preorder

2.Inorder

3.PostOrder

Enter your choice: 1

Preorder:

60 40 69 65 70 77

-----------------------------------

Select operation

1.Create Tree/Insert node

2.Traverse tree

0.Exit

Enter your choice: 2

Traversal Type :

1.Preorder

2.Inorder

3.PostOrder

Enter your choice: 2

Inorder:

40 60 65 69 70 77

-----------------------------------

Select operation

1.Create Tree/Insert node

2.Traverse tree

0.Exit

Enter your choice: 2

Traversal Type :

1.Preorder

2.Inorder

3.PostOrder

Enter your choice: 3

Postorder:

40 65 77 70 69 60

-----------------------------------

Select operation

1.Create Tree/Insert node

2.Traverse tree

0.Exit

Enter your choice: 0