```
/*
Name:Rajesh Alane
Gr No:21820050
Roll no:231070
Assignment 2: Threaded Binary Tree
--------------------------------------------------------------------------------------------
*/

#include<iostream>
using namespace std;

class node //tbt node structure
{
	public:
		int data;	//data of the node
		node *lchild,*rchild;	//links to child nodes
		int lbit,rbit;		//flag indicating link or thread
};

class tbt
{
	node *root;	//dummy node
	node *presuc(node *t);	//finds preorder successor
	node *insuc(node *t);	//finds inorder successor
	public:
		tbt()	//default constructor
		{
			root=new node; //creating dummy node
			root->rbit=1;
			root->lbit=0;
			root->rchild=root->lchild=root; //pointing to self
		}
		void create(int);		//create function
		void inorder();		//inorder function
		void preorder();		//preorder function
};

void tbt::create(int x)
{
	node *p,*parent,*current;
	p=new node;	//creating new node
	p->data=x;  //assigning data value
	if(root->lchild==root)	//dummy points to itself
	{
		parent=root;
		p->lbit=parent->lbit;
		p->rbit=parent->rbit;
		parent->lchild=p;
		parent->lbit=1;		//indicating left link
		p->rbit=0;
```

```
                p->rchild=parent;
        }
        else
        {
                current=root->lchild;    //actual root stored

                while(current!=root)    //looping till current is  not root
                {
                        parent=current;   //parent node
                        if(x<current->data && current->lbit==1)   //moving to leftmost
                        {
                                current=current->lchild;    //attaching to the left
                        }
                        else if(x>current->data && current->rbit==1)
                        {
                                current=current->rchild;     //attaching  to the right
                        }
                        else break;
                }

                if(x<parent->data)
                {
                        p->lbit=parent->lbit;    //copying links
                        p->lchild=parent->lchild;
                        parent->lchild=p;              //linking node to left
                        parent->lbit=1;
                        p->rbit=0;
                        p->rchild=parent;
                }
                else
                {
                        p->rbit=parent->rbit;        //copying links
                        p->rchild=parent->rchild;
                        parent->rchild=p;            //linking node to  right
                        parent->rbit=1;
                        p->lbit=0;
                        p->lchild=parent;
                }
        }
}

node *tbt::presuc(node *t)
{
        if(t->lbit==1)                         //if left child is present
                return(t->lchild);    //return left child
        if(t->rbit==1)                         //if right child is present
                return(t->rchild);    //return right child
        if(t->rbit==0)                       //if right is thread
                t=t->rchild;            //move to right
                return(t->rchild);
```

```
}

node *tbt::insuc(node *t)
{
	if(t->rbit==1)
	{
		t=t->rchild;		//moving to right
		while(t->lbit==1)  //moving to leftmost element
		{
			t=t->lchild;		//moving to left
		}
		return(t);
	}
	else
	{
		return (t->rchild);   //return right child
	}
}

void tbt::inorder()
{
	node *t;
	t=root->lchild;		//stores the root
	while(t->lbit==1)		//goint to leftmost element
	{
		t=t->lchild;
	}
	while(t!=root)		//till reach last element
	{
		cout<<t->data<<"\t";    //print data
		t=insuc(t);			//goint to successor
	}
}

void tbt::preorder()
{
	node *t;
	t=root->lchild;		//stores actual root
	while(t!=root)		//till we reach last element
	{
		cout<<t->data<<"\t";    //print data
		t=presuc(t);			//going to successor
	}
}

int main()
{
	tbt t;
	int x,ch;
	do
	{
		cout<<"\n--------------------- Threaded Binary Tree ------------------------\n";
		cout<<"1:Create\n2:Inorder\n3:Preorder\n4:Exit\n";
		cout<<"Enter your choice:";
		cin>>ch;
		switch(ch)
		{
			case 1:
				cout<<"Enter data to insert:";
				cin>>x;
				t.create(x);   //calling create
				break;
			case 2:
				cout<<"\n\t\t---Inorder Traversal---\n";
				t.inorder();      //inorder traversal
				break;
			case 3:
				cout<<"\n\t\t---Preorder Traversal---\n";
				t.preorder();     //preorder traversal
				break;
		}
	}while(ch!=4);
	return 0;
}

/*
-----------------------------OUTPUT--------------------------

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:50

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:45

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:55

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:35

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:48

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:51

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:1
Enter data to insert:58

--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:2

        ---Inorder Traversal---
35      45      48      50      51      55      58
--------------------- Threaded Binary Tree ------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:3
```

```
                ---Preorder Traversal---
50      45      35      48      55      51      58
---------------------- Threaded Binary Tree -------------------------
1:Create
2:Inorder
3:Preorder
4:Exit
Enter your choice:4
*/
```