Skill Development Lab-II 2018-2019
Name: Omkar Dhaigude
Roll no: 223015
Gr no:17U052

ASSIGNMENT 2

Aim : Construct a threaded binary search tree by inserting values in the given order and traverse it in inorder traversal using threads.

Objective : To understand the concept of binary threading and to understand insertion and inorder traversal on threaded binary tree.

Theory:

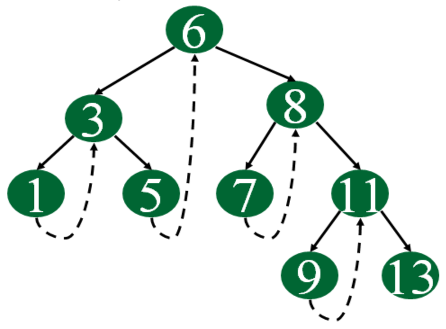
Threaded binary tree : A binary tree is *threaded* by making all right child pointers that would normally be null point to the inorder successor of the node (**if** it exists), and all left child pointers that would normally be null point to the inorder predecessor of the node.

There are two types of threaded binary trees.
Single Threaded: Where a NULL right pointers is made to point to the inorder successor (if successor exists)

Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.

The threads are also useful for fast accessing ancestors of a node.

Following diagram shows an example Single Threaded Binary Tree. The dotted lines represent threads.



Algorithm :

1. Define required structure for threaded binary tree.

```
      struct Node
{
  struct Node *left, *right;
  int info;

  // True if left pointer points to predecessor
  // in Inorder Traversal
  boolean lthread;

  // True if right pointer points to successor
  // in Inorder Traversal
  boolean rthread;
};
```

2. To perform insertion 3 cases we need to understand

Case 1: Insertion in empty tree

Both left and right pointers of tmp will be set to NULL and new node becomes the root.

```
root = tmp;
tmp -> left = NULL;
tmp -> right = NULL;
```

Case 2: When new node inserted as the left child

After inserting the node at its proper place we have to make its left and right threads points to inorder predecessor and successor respectively. The node which was inorder successor. So the left and right threads of the new node will be-

```
tmp -> left = par ->left;
tmp -> right = par;
```

Before insertion, the left pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

```
par -> lthread = false;
par -> left = temp;
```

Case 3: When new node is inserted as the right child
The parent of tmp is its inorder predecessor. The node which was inorder successor of the parent is now the inorder successor of this node tmp. So the left and right threads of the new node will be-

```
tmp -> left = par;
tmp -> right = par -> right;
```

Before insertion, the right pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

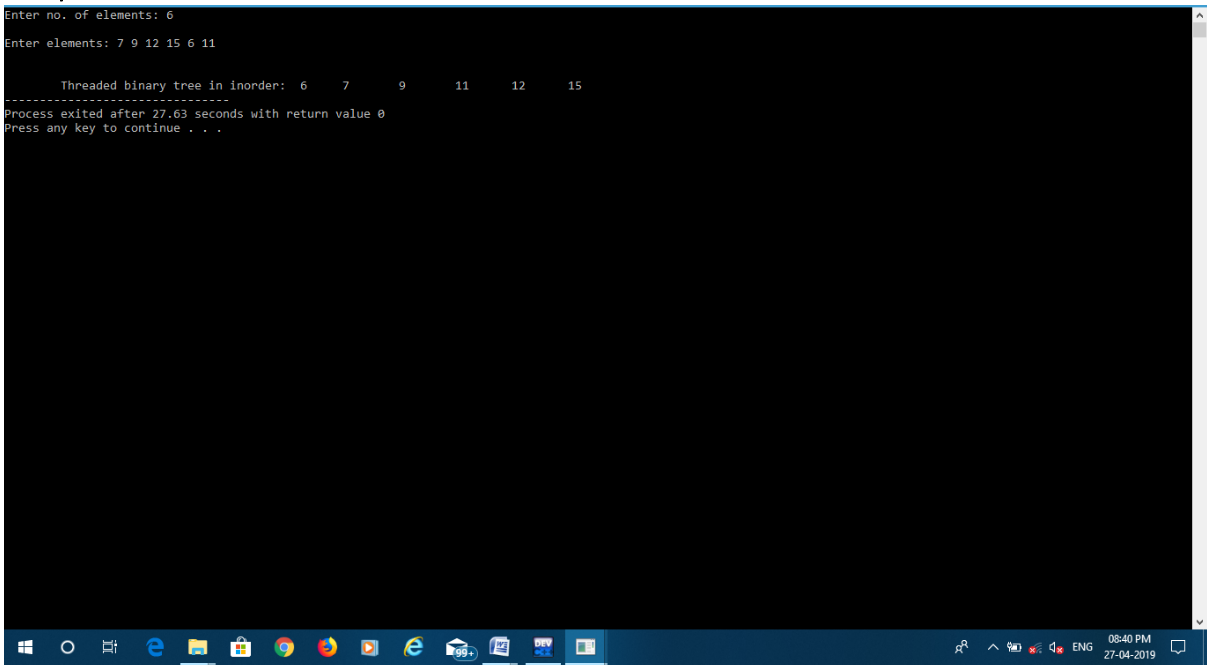
```
par -> rthread = false;
par -> right = tmp;
```

Code:

```
#include<iostream>
using namespace std;

class ttree
{
        private:
                struct thtree
                {
                        int left;
                        thtree *leftchild;
                        int data;
                        thtree *rightchild;
                        int right;
                }*th_head;

        public:
                ttree();
                void insert(int num);
                void inorder();

};

ttree::ttree()
{
        th_head=NULL;
}
void ttree::insert(int num)
{


        thtree *head=th_head,*p,*z;

        z=new thtree;
        z->left=true;
        z->data=num;
        z->right=true;

        if(th_head==NULL)
        {
                head=new thtree;
                head->left=false;
                head->leftchild=z;
                head->data=-9999;
                head->rightchild=head;
                head->right=false;

                th_head=head;
                z->leftchild=head;
                z->rightchild=head;

        }
```

```
else
{
        p=head->leftchild;
        while(p!=head)
        {
                if(p->data > num)
                {
                        if(p->left!=true)
                        p=p->leftchild;
                        else
                        {
                                z->leftchild=p->leftchild;
                                p->leftchild=z;

                                p->left=false;
                                z->right=true;
                                z->rightchild=p;
                                return;
                        }
                }
                else
                {
                        if(p->data < num)
                        {
                                if(p->right!=true)
                                p=p->rightchild;
                                else
                                {
                                        z->rightchild=p->rightchild;
                                        p->rightchild=z;

                                        p->right=false;
                                        z->left=true;
                                        z->leftchild=p;
                                        return;
                                }
                        }
                }
        }
        }
}
void ttree::inorder()
{
        thtree *a;
        a=th_head->leftchild;
        while(a!=th_head)
        {
                while(a->left==false)

                        a=a->leftchild;
                        cout<<a->data<<"\t";



                while(a->right==true)
                {
                        a=a->rightchild;

                        if(a==th_head)
                        break;

                        cout<<a->data<<"\t";
                }
                a=a->rightchild;
        }
}

int main()
{
        ttree th;
        int n,e;
        cout<<"Enter no. of elements: ";
        cin>>n;
        cout<<"\nEnter elements: ";
        for(int i=0;i<n;i++)
        {
                cin>>e;
                th.insert(e);
        }

        cout<<"\n\n\tThreaded binary tree in inorder:  ";
        th.inorder();
}
```

Output:



```
Enter no. of elements: 6

Enter elements: 7 9 12 15 6 11


        Threaded binary tree in inorder:  6     7       9       11      12      15
--------------------------------
Process exited after 27.63 seconds with return value 0
Press any key to continue . . .
```

Conclusion : We successfully implement threaded binary tree and peform insertion and inorder traversal operation on it.