

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

ASSIGNMENT NO.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 study the concept of threaded binary tree.

Theory:-

A **binary tree** can be represented by using array representation or linked list representation. When a binary tree is represented using linked list representation. If any node is not having a child we use a NULL pointer. These special pointers are threaded and the binary tree having such pointers is called a threaded binary tree. Thread in a binary tree is represented by a dotted line. In linked list representation of a binary tree, they are a number of a NULL pointer than actual pointers. This NULL pointer does not play any role except indicating there is no child. The **threaded binary tree** is proposed by A.J Perlis and C Thornton. There are three ways to thread a binary tree.

1. In the in order traversal When The right NULL pointer of each leaf node can be replaced by a thread to the successor of that node then it is called a right thread, and

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

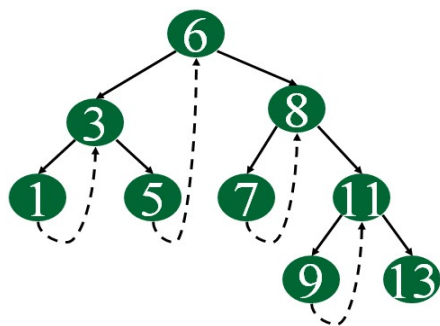
GR NO:- 17U208

the resultant tree called a right threaded tree or right threaded binary tree.

2. When The left NULL pointer of each node can be replaced by a thread to the predecessor of that node under in order traversal it is called left thread and the resultant tree will call a left threaded tree.
3. In the in order traversal, the left and the right NULL pointer can be used to point to predecessor and successor of that node respectively. then the resultant tree is called a fully threaded tree.

In the threaded binary tree when there is only one thread is used then it is called as one way threaded tree and when both the threads are used then it is called the two way threaded tree. The pointer point to the root node when If there is no in-order predecessor or in-order successor.

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



Algorithm:-

Let **tmp** be the newly inserted node. There can be three cases during insertion:

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;
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
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;
```

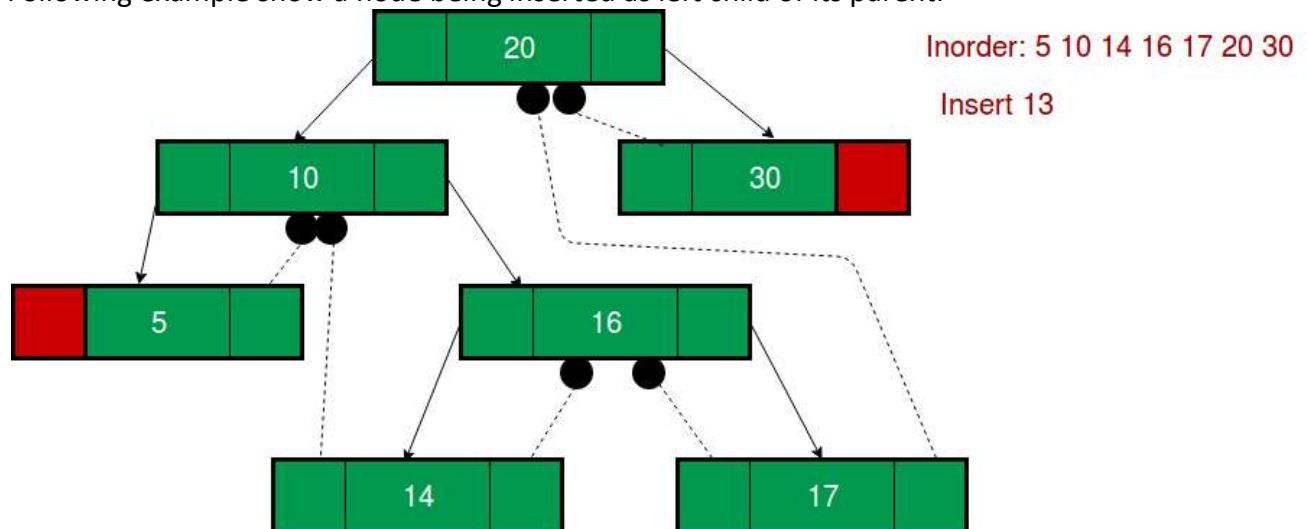
```
tmp -> right = par;
```

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;
```

Following example show a node being inserted as left child of its parent.



After insertion of 13,

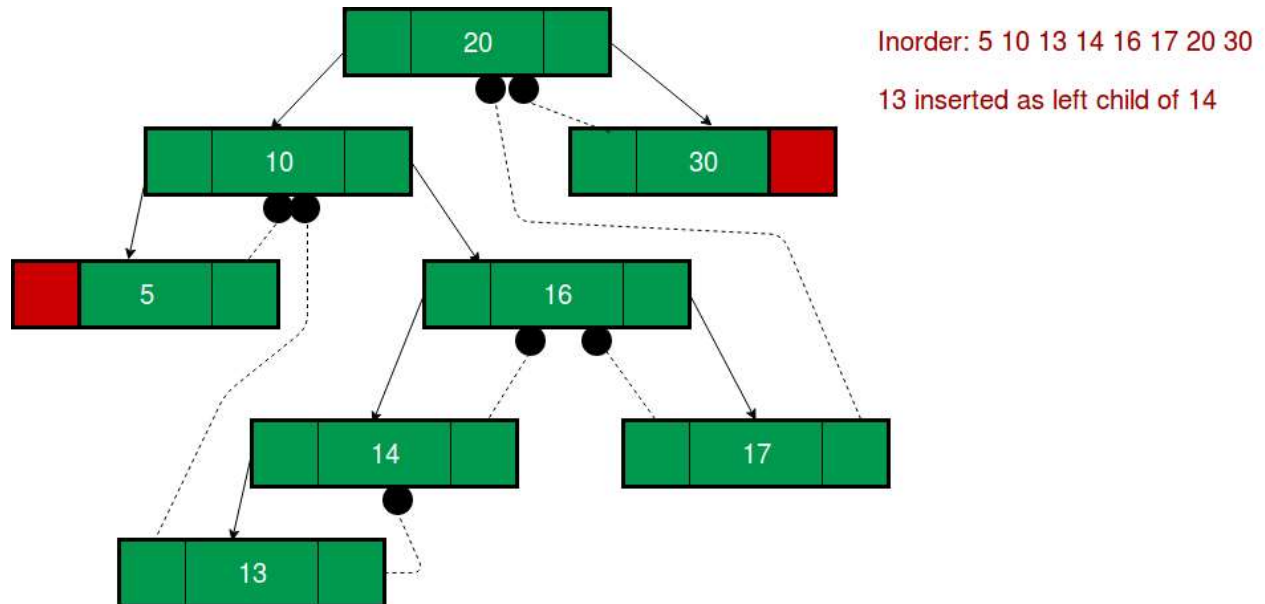
NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208



Predecessor of 14 becomes the predecessor of 13, so left thread of 13 points to 10.

Successor of 13 is 14, so right thread of 13 points to left child which is 13.

Left pointer of 14 is not a thread now, it points to left child which is 13.

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;
```

Program Code:-

NAME:-RISHIKESH SHEDE
CLASS:- SYBTECH-C
BATCH:- C-2
ROLL NO:- 223053
GR NO:- 17U208

```
#include <iostream>

using namespace std;

class TBT;

class node
{
    node *left,*right;

    int data;

    bool rbit,lbit;

public:
    node()
    {
        left=NULL;

        right=NULL;

        rbit=lbit=0;
    }

    node(int d)
    {
        left=NULL;

        right=NULL;

        rbit=lbit=0;
    }
}
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
        data=d;

    }

    friend class TBT;

};

class TBT

{

    node *root; //acts as a dummy node

public:

    TBT() //dummy node initialization

    {

        root=new node(9999);

        root->left=root;

        root->rbit=1;

        root->lbit=0;

        root->right=root;

    }

    void create();

    void insert(int data);

    node *inorder_suc(node *);

    void inorder_traversal();
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
        node * preorder_suc(node *c);

        void preorder_traversal();

};

//-----

void TBT::preorder_traversal()
{
    node *c=root->left;
    while(c!=root)
    {
        cout<<" "<<c->data;
        c=preorder_suc(c);
    }
}

void TBT::inorder_traversal()
{
    node *c=root->left;
    while(c->lbit==1)
        c=c->left;
    while(c!=root)
    {
        cout<<" "<<c->data;
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
        c=inorder_suc(c);
    }
}

node* TBT::inorder_suc(node *c)
{
    if(c->rbit==0)
        return c->right;
    else
        c=c->right;
    while(c->lbit==1)
    {
        c=c->left;
    }
    return c;
}

node *TBT::preorder_suc(node *c)
{
    if(c->lbit==1)
    {
        return c->left;
    }
}
```


NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
        while(c->rbit==0)

        {

                c=c->right;

        }

        return c->right;

}

//----- Create Method

void TBT::create()

{

        int n;

        if(root->left==root&&root->right==root)

        {

                cout<<"\nEnter number of nodes:";

                cin>>n;

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

                {

                        int info;

                        cout<<"\nEnter data: ";

                        cin>>info;

                        this->insert(info);
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
    }
    }
    else
    {
        cout<<"\nTree is Already created.\n";
    }
}

void TBT::insert(int data)
{
    if(root->left==root&&root->right==root) //no node in tree
    {
        node *p=new node(data);
        p->left=root->left;
        p->lbit=root->lbit; //0
        p->rbit=0;
        p->right=root->right;
        root->left=p;
        root->lbit=1;
        cout<<"\nInserted start"<<data;
        return;
    }
}
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
    }

    node *cur=new node;

    cur=root->left;

    while(1)

    {

        if(cur->data<data) //insert right

        {

            node *p=new node(data);

            if(cur->rbit==0)

            {

                p->right=cur->right;

                p->rbit=cur->rbit;

                p->lbit=0;

                p->left=cur;

                cur->rbit=1;

                cur->right=p;

                //cout<<"\nInserted right "<<data;

                cout<< data<<" Inserted right"<<" of "<< cur->data<<endl;

                return;

            }

        }

    }
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
        else

            cur=cur->right;

        }

        if(cur->data>data) //insert left
        {

            node *p=new node(data);

            if(cur->lbit==0)

            {

                p->left=cur->left;

                p->lbit=cur->lbit;

                p->rbit=0;

                p->right=cur; //successor

                cur->lbit=1;

                cur->left=p;

                cout<<data <<" Inserted left "<<" of "<<cur->data<<endl;

                return;

            }

            else

                cur=cur->left;

        }

    }
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

}

```
int main() {  
    TBT t1;  
  
    int value;  
  
    int choice;  
  
    do  
    {  
  
        cout<<"\n1.Create Tree\n2.Insert into  
tree\n3.Preorder\n4.Inorder\n0.Exit\nEnter your choice: ";  
  
        cin>>choice;  
  
        switch(choice)  
        {  
  
            case 1:  
  
                t1.create();  
  
                break;  
  
            case 2:  
  
                cout<<"\nEnter Number(data): ";  
  
                cin>>value;
```

NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

```
                t1.insert(value);

                break;

            case 3:

                cout<<"\nPreorder traversal of TBT\n";

                t1.preorder_traversal();

                break;

            case 4:

                cout<<"\nInoder Traversal of TBT\n";

                t1.inorder_traversal();

                break;

            default:

                cout<<"\nWrong choice";

        }

    }while(choice!=0);

    return 0;

}
```

NAME:-RISHIKESH SHEDE

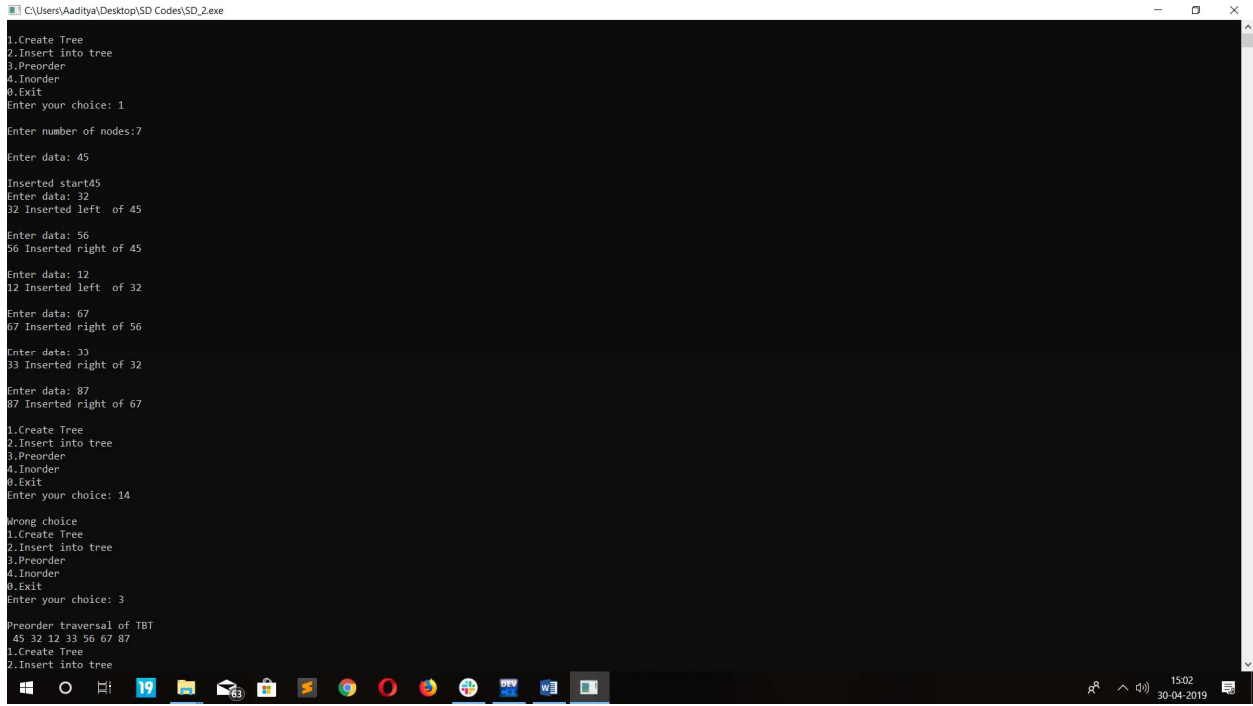
CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

Output Screenshots:-



```
C:\Users\Aaditya\Desktop\SD Codes\SD_2.exe
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 1
Enter number of nodes:7
Enter data: 45
Inserted start45
Enter data: 32
32 Inserted left of 45
Enter data: 56
56 Inserted right of 45
Enter data: 12
12 Inserted left of 32
Enter data: 67
67 Inserted right of 56
Enter data: 33
33 Inserted right of 32
Enter data: 87
87 Inserted right of 67

1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 14
Wrong choice
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 3
Preorder traversal of TBT
45 32 12 33 56 67 87
1.Create Tree
2.Insert into tree
```

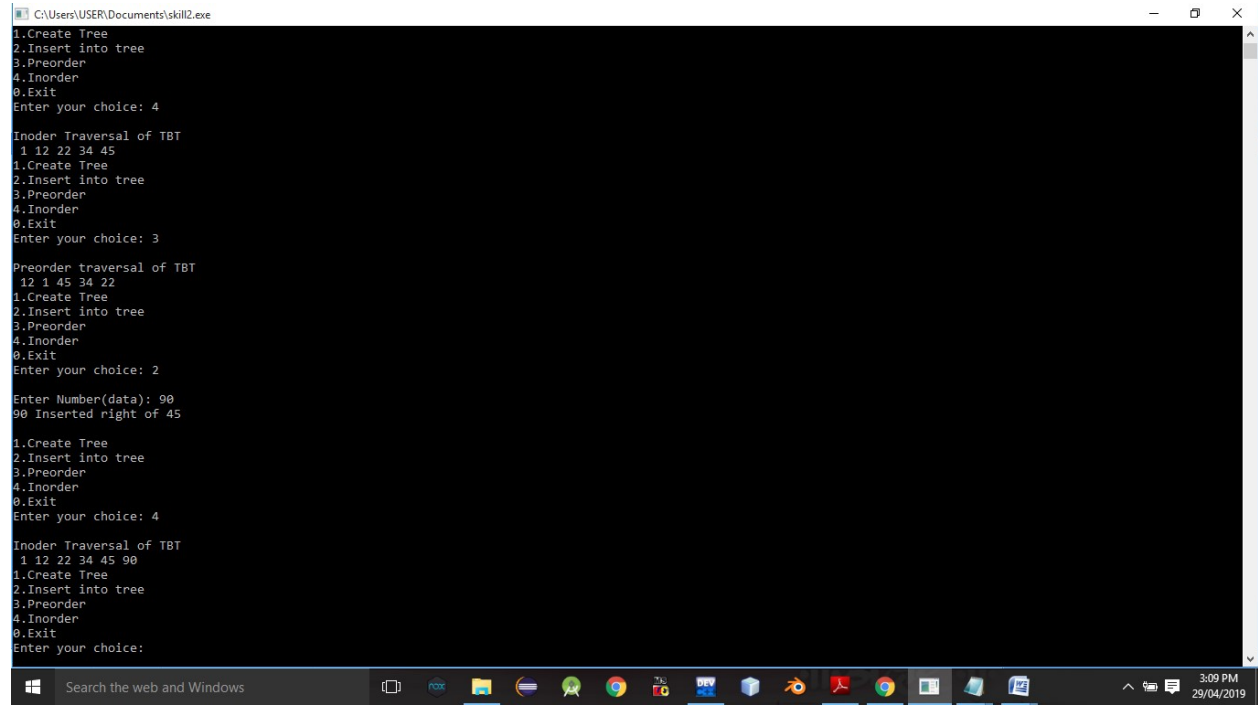
NAME:-RISHIKESH SHEDE

CLASS:- SYBTECH-C

BATCH:- C-2

ROLL NO:- 223053

GR NO:- 17U208

A screenshot of a Windows desktop environment. The taskbar at the bottom shows various application icons including File Explorer, Edge, and several utility programs. The system clock indicates 3:09 PM on 29/04/2019. A terminal window titled 'C:\Users\USER\Documents\skill2.exe' is open, displaying the output of a threaded binary tree program. The program menu includes: 1.Create Tree, 2.Insert into tree, 3.Preorder, 4.Inorder, and 0.Exit. The user has entered choice 4, resulting in an Inorder Traversal of TBT showing the sequence 1 12 22 34 45. The user then enters choice 3, resulting in a Preorder traversal of TBT showing 12 1 45 34 22. Next, the user enters choice 2, and the program prompts for a number (90) to be inserted to the right of 45. Finally, the user enters choice 4 again, resulting in an Inorder Traversal of TBT showing 1 12 22 34 45 90. The program menu is displayed again at the bottom of the terminal output.

```
C:\Users\USER\Documents\skill2.exe
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 4

Inoder Traversal of TBT
1 12 22 34 45
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 3

Preorden traversal of TBT
12 1 45 34 22
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 2

Enter Number(data): 90
90 Inserted right of 45

1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice: 4

Inoder Traversal of TBT
1 12 22 34 45 90
1.Create Tree
2.Insert into tree
3.Preorder
4.Inorder
0.Exit
Enter your choice:
```

Conclusion:- Thus, we have studied Threaded binary tree.