Shrikrushna S Zirape
21286

PICT, PUNE

Assignment NO-3

Problem Statement-

Creat a inorder threaded binary tree and perform inorder and preorder traversal Aanlyze time and space complexity of an algorithm.

Objective:-

To understand concept of TBT and it's properties.

To perform operations such as inorder and preorder traversal.

Outcome:

To impliment TBT and perform different operations on it.

To write menu driven and modular program in C++.

To impliment Concept of OOP in C++.

S/W Requirement:

Windows 10

Eclipse IDE

Gcc Compiler.

H/W Requirement:

- 64 bit OS.
- intel is 8th gen

Theory:-

- In Binary Tree there are many nodes that have an empty left child or empty right child or both.

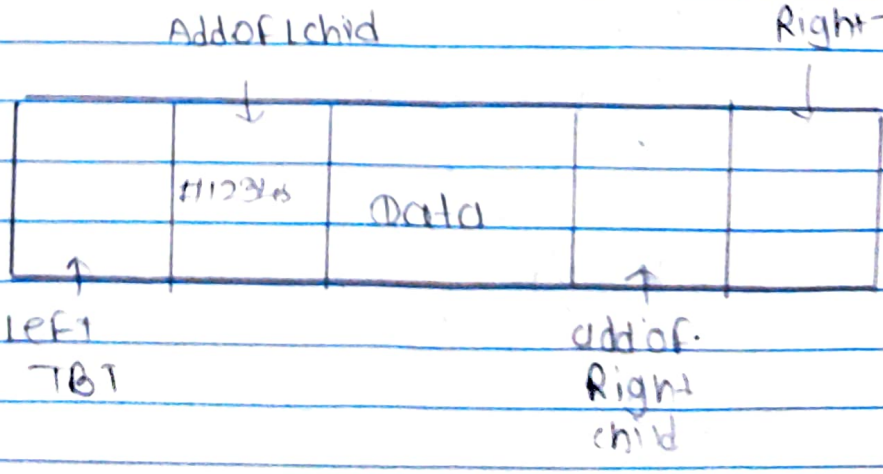
- you can utilize these fields such that empty left child or a node points to it's inorder predessor and empty right child of a node points to it's inorder successor in inorder Traversal.

- The Binary tree stores such information in null pointer are threaded binary tree.

Types:-

A] Single Threaded Tree
B] Double Threaded Tree

Representing a TBT



PICT, PUNE

Pseudo Code:

A]
```
class Node {
    int data;
    Node * lchild, *rchild;
    bool lthread, rthread;
    Public :
        Node() {
            lchild := rchild := NULL;
            lthred := rthreld := 0; data := 0;
        }
        Node (int x) {
            lchild := rchild := NULL;
            lthred := rthred := 0;
            data := x;
        }
    Friend class TBT;
}
```

B]
```
Class TBT {
    Node * root, *header;
    Public;
        TBT() {
            root := header := NULL;
        }
        Void insert (int);
        Void inorder ();
        Void preorder();
}
```

(i)

```
Void TBT :: insert (int K)
  if (root == NULL) then.
      header := New Node (-99);
      header → rchild = header;
      root := New Node (K);
      root → lchild := header;
      root → rchid := header;
      header → lchid = root;
      return;
  Node * temp = root;
  Node * newnode = new Node (K);
  While(1) do {
     if (K < temp → data) then
        if (temp → lchild lbhred {
            temp := temp → lchid; }
        else.
            newnode → lchid := temp → lchild;
            newnode → rchid := temp
            temp → lbhred := true;
            temp → lchild = temp;
            return
          }
      }
    if (temp → rthread)
    {
       temp := temp → rchild;
    }
   else {
```

```
newnode->lchild := temp;
newnode->rchild := temp->rchild;
temp->rchild := temp newnode;
temp->rthread := true;
return.
}
}}}
```

(ii)

```
Void TBT: inorder() {
  Node * temp
  While (temp->lthread) do
    temp := temp->lchild;
  While (temp != header) do {
    print (temp->data)
    if (temp->rthread) {
      temp := temp->rchild.
      While (temp->lthred) {
        temp := temp->lchild.
      }
    }
    else {
      temp := temp->rchild;
    }
}}
```

(iii)

```
Void TBT : Preorder ()
{
Node *temp := root;
while (temp != header) do{
    While (temp -> lthread) {
        Print (temp->data);
        temp := temp -> lchild;
    }
    Print (temp->data)
    While (temp -> rthread) do {
        temp := temp -> rchild;
        if (temp == header{
            return;
        }
    } temp := temp -> rchild;
} }
```

Test Cases :-

NO.	input	output	expec. output	Result
	Enter key : 10			
1	Enter key : 9			
	Enter key : 5			
	Enter key : 3			
	Enter key : 4			
	choice : 2	inorder : 3,4,5,9,10	inorder 3,4,5,9,10	Pass
	choice : 3	preorder : 10,9,5,3,4	preorder 10,9,5,3,4	Pass

Conclusion :-

Thus He implimented the TBT and traversal (inorder and preorder) successfully.