Practical-5(B-10)

**Problem Statement:**

Consider threading a binary tree using preorder threads rather than inorder threads. Design an algorithm for traversal without using stack and analyze its complexity.

**Code:**

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

class Tnode
{
   public:
      int data;
      Tnode *left;
      Tnode *right;
      int lbit, rbit;
};

class TBT
{
   Tnode *root,*head;
   public:
   TBT()
   {
      root = NULL;
      head = NULL;
   }
   void createTBT();
   void preorder();

};

void TBT::preorder() // preorder traversal of TBT.
{
   Tnode *temp;
   temp = root;
   int flag =0;

   if(root == NULL)
   {
      cout<<"\nTree is empty";
   }
   else
   {
      while(temp != head)
      {
         if(flag ==0)
         {
            cout<<" "<<temp ->data;
         }
         if(temp->lbit==1 && flag ==0)
         {
            temp = temp ->left;

         }
         else
         {
            if(temp ->rbit == 1)
            {
               temp = temp ->right;
               flag=0;
            }
            else
```

```
            {
               temp = temp->right;
               flag =1;
            }
         }

      }
   }
}


void TBT::createTBT()
{
   int flag = 0;
   char ans;
   Tnode *new_node, *temp;
   head = new Tnode(); // allocation of memory for head.
   head ->data = -1;
   head ->left = head;
   head ->right = head;
   head ->lbit = 0;
   head ->rbit = 0;
   root =  new Tnode(); // allocation of memory for root.
   cout<<"Enter root node : ";
   cin>> root->data;
   head ->left = root;
   head -> lbit = 1;
   root ->left = head;
   root ->right = head;
   root ->lbit = 0;
   root ->rbit = 0;

   do
   {
      new_node = new Tnode();
      cout<<"\nEnter new node : ";
      cin>>new_node ->data;
      new_node ->lbit = 0;
      new_node ->rbit = 0;
      temp = root;
      flag = 0;
      while(flag ==0)  // find proper place for new node.
      {
         if(new_node -> data < temp -> data)
         {
            if(temp ->lbit == 0)
            {
               new_node ->left = temp ->left;
               temp ->left = new_node;
               temp->lbit=1;
               new_node ->right = temp;
               flag = 1;
            }
            else
            {
               temp = temp ->left;
            }
         }
         else if(new_node -> data > temp ->data)
         {
            if(temp ->rbit ==0)
            {
               new_node ->right = temp ->right;
               temp ->right  = new_node;
               temp ->rbit = 1;
               new_node ->left = temp;
```

```
                flag =1;
            }
            else
            {
                temp=temp->right;
            }
        }
        else
        {
            cout<<"\n Data is already exist";
        }

    }

    cout<<"\nDo you want to continue : ";
    cin>>ans;
  }while(ans=='Y' || ans=='y');
}

int main()
{
  TBT T;
  T.createTBT();

 // Preorder Display.
  T.preorder();

  return 0;
}
```

**OUTPUT:**

```
Enter root node : 10
Enter new node : 5
Do you want to continue : y
Enter new node : 15
Do you want to continue : y
Enter new node : 2
Do you want to continue : y
Enter new node : 8
Do you want to continue : y
Enter new node : 12
Do you want to continue : y
Enter new node : 20
Do you want to continue : n
 10 5 2 8 15 12 20
```