Week -9

Q1

AIM: Write a menu driven program to implement Binary tree with the following operations.

i. Insertion
ii. Preorder
iii. Inorder
iv. Postorder

Description:

1. START
2. Create a node class for having left and right attributes for each object of binary tree class
3. Now create a binary tree class and define the required methods as mentioned in the given problem.

Insertion:

```
i.     START
ii.    Enter the value to be inserted in data variable.
iii.       if self.root is None:
iv.            self.root=Node(data)
v.         else:
vi.            ptr=self.root
vii.           n=int(input("\n1. Left \t 2. Right\nEnter which side u want
       to insert: "))
viii.          while (ptr.left is not None) or (ptr.right is not None):
ix.
x.                 if n==1 and ptr.left is not None:
xi.                    ptr=ptr.left
xii.                   n=int(input("\n1. Left \t 2. Right\nEnter which side u
       want to insert: "))
xiii.              elif n==2 and ptr.right is not None:
xiv.                   ptr=ptr.right
xv.                    n=int(input("\n1. Left \t 2. Right\nEnter which side u
       want to insert: "))
xvi.               elif (n==1 and ptr.left is None) or (n==2 and ptr.right is
       None):
xvii.                  break
```

```
xviii.          else:
xix.                print("Wrong choice, try again")
xx.         newnode=Node(data)
xxi.        if n==1:
xxii.           ptr.left=newnode
xxiii.      elif n==2:
xxiv.           ptr.right=newnode
xxv.   STOP
```

Preorder Traversal

```
i.     def preorder(self,root):
ii.        if (root):
iii.           print(root.data,end=' ')
iv.            self.preorder(root.left)
v.             self.preorder(root.right)
```

Inorder Traversal

```
i.     def inorder(self,root):
ii.        if (root):
iii.           self.inorder(root.left)
iv.            print(root.data,end=' ')
v.             self.inorder(root.right)
```

Postorder Traversal

```
i.     def postorder(self,root):
ii.        if (root):
iii.           self.postorder(root.left)
iv.            self.postorder(root.right)
v.             print(root.data,end=' ')
```

4. Now outside the class, create a binary tree object and do the given operations as required using a while loop
5. STOP

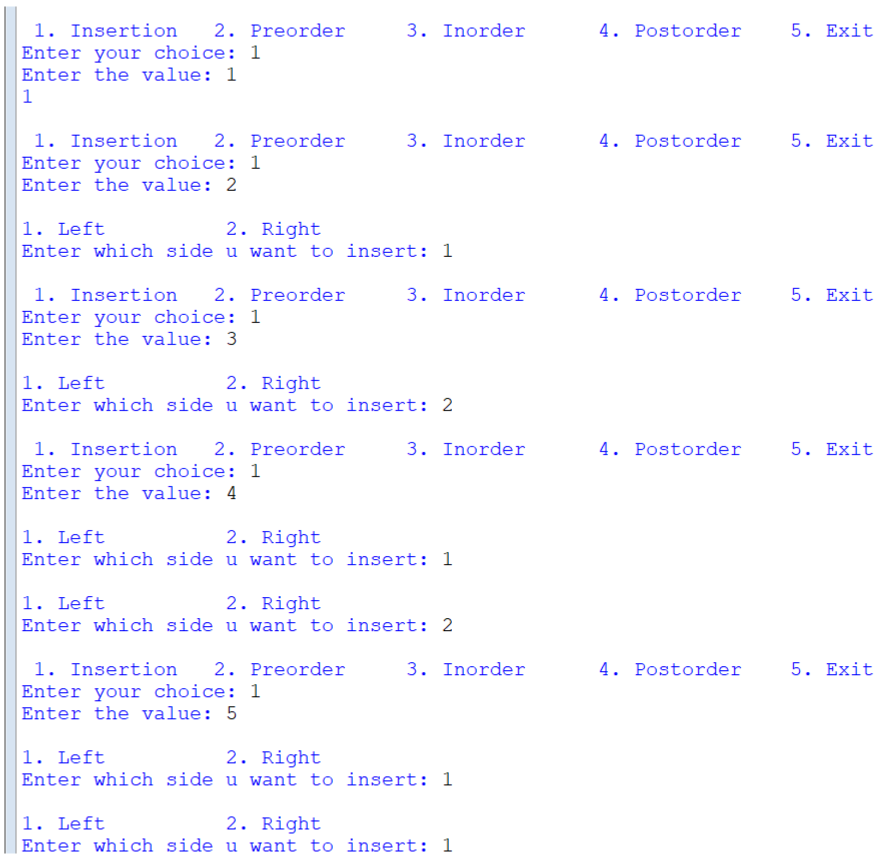
Program:

```
class Node:
  def __init__(self,data):
    self.data=data
    self.left=None
    self.right=None
class Binary_tree:
  def __init__(self):
        self.root=None
    def insertion(self):
        data=int(input("Enter the value: "))
        if self.root is None:
            self.root=Node(data)
            print(self.root.data)
        else:
            ptr=self.root
            n=int(input("\n1. Left \t 2. Right\nEnter which side u want to insert: "))
            while (ptr.left is not None) or (ptr.right is not None):


                if n==1 and ptr.left is not None:
                    ptr=ptr.left
                    n=int(input("\n1. Left \t 2. Right\nEnter which side u want to
insert: "))
                elif n==2 and ptr.right is not None:
                    ptr=ptr.right
                    n=int(input("\n1. Left \t 2. Right\nEnter which side u want to
insert: "))
                elif (n==1 and ptr.left is None) or (n==2 and ptr.right is None):
                    break
                else:
                    print("Wrong choice, try again")
            newnode=Node(data)
            if n==1:
                ptr.left=newnode
            elif n==2:
                ptr.right=newnode
    def preorder(self,root):
        if (root):
            print(root.data,end=' ')
            self.preorder(root.left)
            self.preorder(root.right)
    def inorder(self,root):
        if (root):
            self.inorder(root.left)
            print(root.data,end=' ')
            self.inorder(root.right)
    def postorder(self,root):
        if (root):
            self.postorder(root.left)
            self.postorder(root.right)
            print(root.data,end=' ')


bt=Binary_tree()
while True:
    n=int(input("\n 1. Insertion\t2. Preorder\t3. Inorder\t4. Postorder\t5. 
Exit\nEnter your choice: "))
    if n==1:
        bt.insertion()
    elif n==2:
        print("The preorder traversal is: ")
        bt.preorder(bt.root)
    elif n==3:
        print("The Inorder traversal is: ")
        bt.inorder(bt.root)
    elif n==4:
        print("The postorder traversal is: ")
        bt.postorder(bt.root)
    elif n==5:
        exit()
    else:
        print("Wrong choice, try again")
```

Output:



```
 1. Insertion   2. Preorder     3. Inorder      4. Postorder    5. Exit
Enter your choice: 1
Enter the value: 1
1

 1. Insertion   2. Preorder     3. Inorder      4. Postorder    5. Exit
Enter your choice: 1
Enter the value: 2

1. Left         2. Right
Enter which side u want to insert: 1

 1. Insertion   2. Preorder     3. Inorder      4. Postorder    5. Exit
Enter your choice: 1
Enter the value: 3

1. Left         2. Right
Enter which side u want to insert: 2

 1. Insertion   2. Preorder     3. Inorder      4. Postorder    5. Exit
Enter your choice: 1
Enter the value: 4

1. Left         2. Right
Enter which side u want to insert: 1

1. Left         2. Right
Enter which side u want to insert: 2

 1. Insertion   2. Preorder     3. Inorder      4. Postorder    5. Exit
Enter your choice: 1
Enter the value: 5

1. Left         2. Right
Enter which side u want to insert: 1

1. Left         2. Right
Enter which side u want to insert: 1
```

Time complexity:

Insertion :O(log(n)) to base 2

Conclusion: The code is error free and it runs as expected