PROGRAM 4 : Beginning with an empty binary search tree, Construct binary search tree by inserting the values in the order given. After constructing a binary tree –

1. Insert new node,
2. Find number of nodes in longest path from root,
3. Minimum data value found in the tree,
4. Change a tree so that the roles of the left and right pointers are swapped at every node,

Search a value

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.L = None

self.R = None

class BST:

def \_\_init\_\_(self):

self.root = None

self.count = 0

def create(self):

while True:

data = int(input("Enter the data: "))

temp = Node(data)

if self.root is None:

self.root = temp

else:

self.insert(self.root, temp)

self.count += 1

ans = input("Do you want to insert more values? (y/n): ")

if ans.lower() != 'y':

break

print(f"The total number of nodes are: {self.count}")

def insert(self, root, temp):

if temp.data > root.data:

if root.R is None:

root.R = temp

else:

self.insert(root.R, temp)

else:

if root.L is None:

root.L = temp

else:

self.insert(root.L, temp)

def disin(self, root):

if root is not None:

self.disin(root.L)

print(root.data, end="\t")

self.disin(root.R)

def dispre(self, root):

if root is not None:

print(root.data, end="\t")

self.dispre(root.L)

self.dispre(root.R)

def dispost(self, root):

if root is not None:

self.dispost(root.L)

self.dispost(root.R)

print(root.data, end="\t")

def search(self, root, key):

temp = root

while temp is not None:

if key == temp.data:

print("KEY FOUND")

return

if key > temp.data:

temp = temp.R

else:

temp = temp.L

print("KEY NOT FOUND")

def height(self, root):

if root is None:

return 0

elif root.L is None and root.R is None:

return 0

else:

return 1 + max(self.height(root.L), self.height(root.R))

def min(self, root):

temp = root

while temp.L is not None:

temp = temp.L

print(temp.data)

def mirror(self, root):

if root is not None:

self.mirror(root.L)

self.mirror(root.R)

root.L, root.R = root.R, root.L

# Main program

t = BST()

while True:

print("\n1) Insert new node 2) Number of nodes in longest path 3) Minimum 4) Mirror 5) Search 6) Inorder 7) Preorder 8) Postorder")

ch = int(input("Choose an option: "))

if ch == 1:

t.create()

elif ch == 2:

print(f"\nNumber of nodes in longest path: {1 + t.height(t.root)}")

elif ch == 3:

print("\nThe min element is: ", end="")

t.min(t.root)

elif ch == 4:

t.mirror(t.root)

print("\nThe mirror of the tree is: ")

t.disin(t.root)

elif ch == 5:

key = int(input("Enter your key: "))

t.search(t.root, key)

elif ch == 6:

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

t.disin(t.root)

elif ch == 7:

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PREORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

t.dispre(t.root)

elif ch == 8:

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POSTORDER\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

t.dispost(t.root)

else:

print("Invalid option. Try again.")

ans = input("\nDo you want to continue? (y/n): ")

if ans.lower() != 'y':

break