# Lab Assignment 5

Name: Taskia Siddika

ID:20301417

Section:03

# Task 1:

def convert\_mirror(root):

#To Do

if root:

location = root.left

root.left = root.right

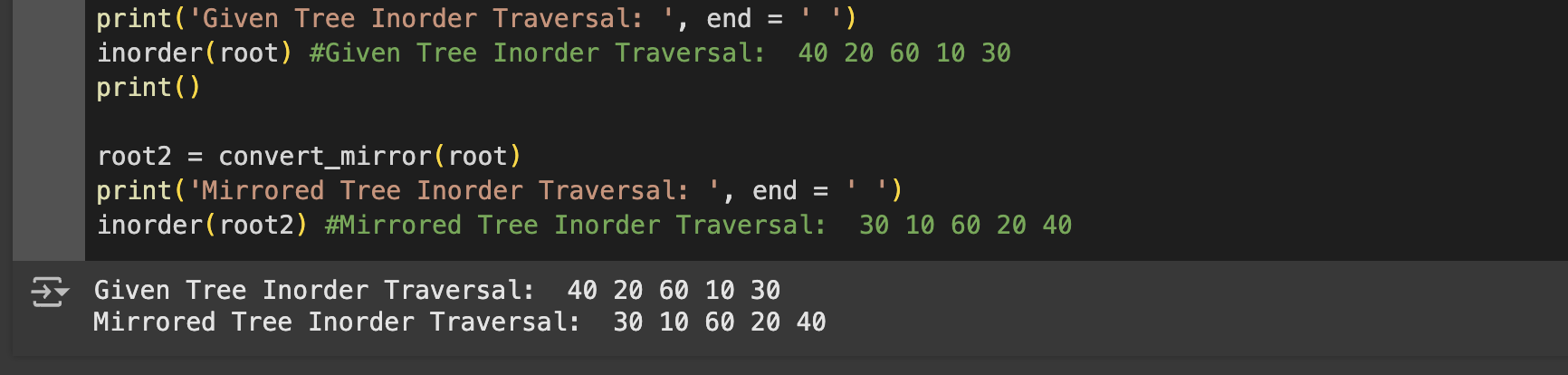
root.right = location

convert\_mirror(root.left)

convert\_mirror(root.right)

return root

**# Output Screenshot**



# 

# 

# 

# 

# 

# 

# Task 2:

def smallest\_level(root, level=None, i=0):

if level == None:

level = {}

if root == None:

return level

if i in level:

level\_now = level[i]

else:

level\_now = None

if level\_now == None:

level[i] = root.elem

else:

node\_now = root.elem

if node\_now < level\_now:

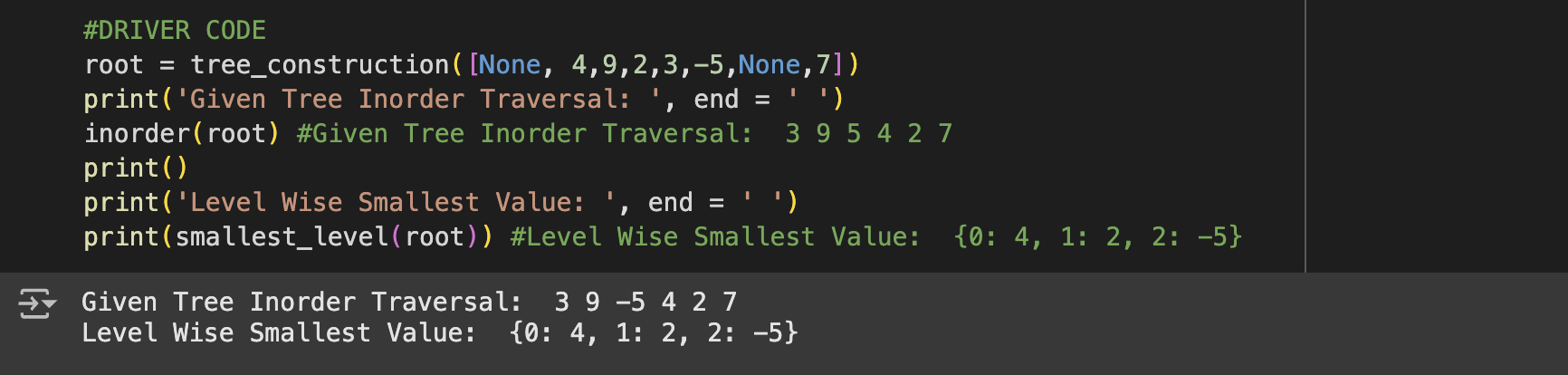
level[i] = node\_now

smallest\_level(root.left, level, i + 1)

smallest\_level(root.right, level, i + 1)

return level

**# Output Screenshot**



# Task 3:

def sumTree(root):

l\_sum = h\_Sum(root.left)

r\_sum = h\_Sum(root.right)

return root.elem + l\_sum + r\_sum

def h\_Sum(root, lvl=1):

if root is None:

return 0

c = root.elem % lvl

if (root.left == None) and (root.right == None):

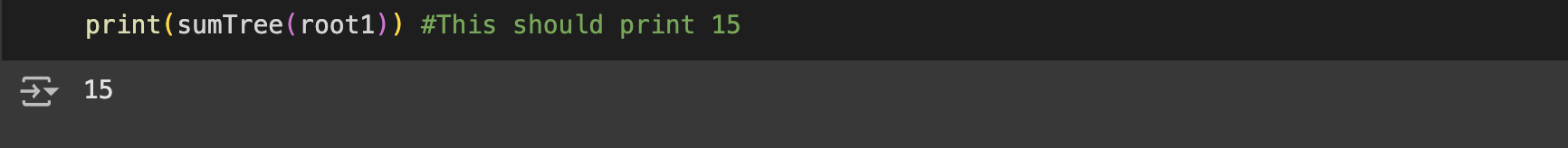
return c

l = h\_Sum(root.left, lvl + 1)

r = h\_Sum(root.right, lvl + 1)

return c + l + r

**# Output Screenshot**

****

# Task 4:

def swap\_child(root, level, M):

if root is None:

return root

c\_lvl = level

m\_lvl = M

if m\_lvl == 0:

l\_node = root.left

r\_node = root.right

root.left = r\_node

root.right = l\_node

return root

if (c\_lvl >= m\_lvl) or (root == None):

return

n\_lvl = c\_lvl + 1

l\_child = swap\_child(root.left, n\_lvl, m\_lvl)

r\_child = swap\_child(root.right, n\_lvl, m\_lvl)

l\_temp = root.left

r\_temp = root.right

root.left = r\_temp

root.right = l\_temp

return root

#Driver Code

root=BTNode('A')

Node1 = BTNode('B')

Node2 = BTNode('C')

Node3 = BTNode('D')

Node4 = BTNode('E')

Node5 = BTNode('F')

Node6 = BTNode('G')

Node7 = BTNode('H')

Node8 = BTNode('I')

Node9 = BTNode('J')

root.left = Node1

root.right = Node2

Node1.left = Node3

Node1.right = Node4

Node2.left = None

Node2.right = Node5

Node3.left = Node6

Node3.right = Node7

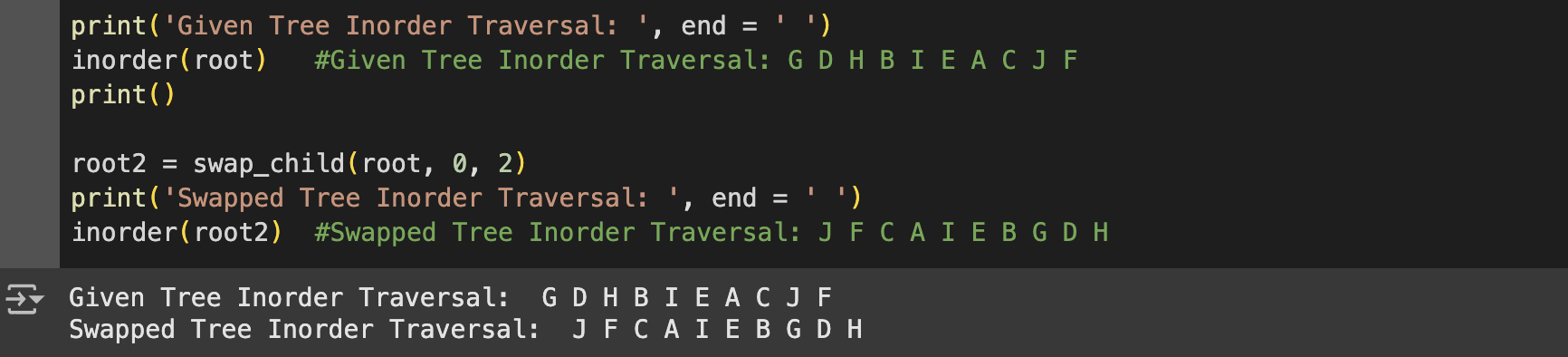
Node4.left = Node8

Node4.right = None

Node5.left = Node9

Node5.right = None

#output screenshot



TASK 5:

def subtract\_summation(root):

if root is None:

return 0

l\_sum = jogfol(root.left)

r\_sum = jogfol(root.right)

return l\_sum - r\_sum

def jogfol(root):

if root is None:

return 0

final = root.elem

final += jogfol(root.left)

final += jogfol(root.right)

return final

#Driver Code

root = BTNode(71)

Node1 = BTNode(27)

Node2 = BTNode(80)

Node3 = BTNode(75)

Node4 = BTNode(87)

Node5 = BTNode(56)

Node6 = BTNode(62)

Node7 = BTNode(41)

Node8 = BTNode(3)

Node9 = BTNode(19)

Node10 = BTNode(89)

root.left = Node1

root.right = Node6

Node1.left = Node2

Node1.right = Node3

Node2.left = Node4

Node2.right = Node5

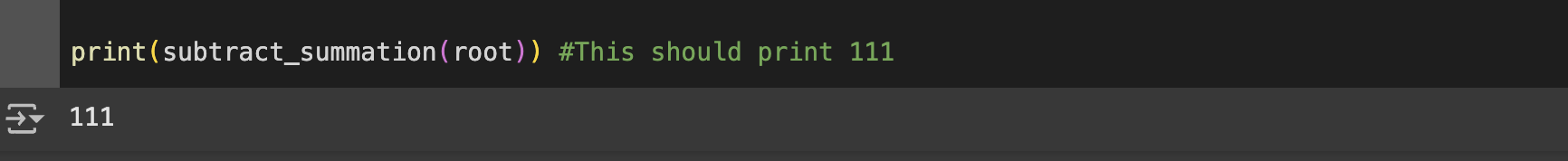
Node6.left = Node7

Node6.right = Node8

Node8.left = Node9

Node8.right = Node10

#output screenshot



TASK 6:

def level\_sum(root, level=1):

if root is None:

return 0

curr = level

curr = root.elem

lchild = level\_sum(root.left, curr + 1)

rchild = level\_sum(root.right, curr + 1)

if curr % 2 == 0: # Even level

total = curr + lchild + rchild

else: # Odd level

total = -curr + lchild + rchild

return total

#output screenshot

