Nama : Tasya Farah Putri A
Kelas : D
NIM : L2001701476

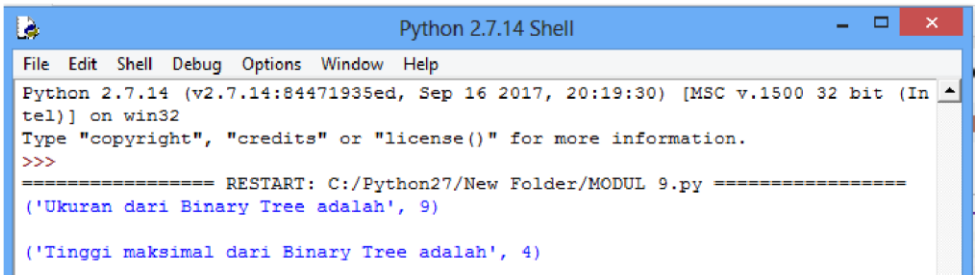
PRAKTIKUM ALGORITMA STRUKTUR DATA – MODUL 9

1. Membuat program pohon biner

```
class simpulbiner(object):
    def __init__(self, data):
        self.data=data
        self.kiri=None
        self.kanan=None

    def __str__(self):
        return str(self.data)

A=simpulbiner('Magetan')
B=simpulbiner('Ngawi')
C=simpulbiner('Madiun')
D=simpulbiner('Ponorogo')
E=simpulbiner('Solo')
F=simpulbiner('Jombang')
G=simpulbiner('Karanganyar')
H=simpulbiner('Pacitan')
I=simpulbiner('Bojonegoro')
J=simpulbiner('Nganjuk')

A.kiri=B; A.kanan=C
B.kiri=D; B.kanan=E
C.kiri=F; C.kanan=G
E.kiri=H
G.kanan=I

datalist=[A.data, B.data, C.data, D.data, E.data, F.data,
          G.data, H.data, I.data, J.data]
level=[]

def preord(sub):
    if sub is not None:
        print(sub.data)
        preord(sub.kiri)
        preord(sub.kanan)
def inord(sub):
    if sub is not None:
        inord(sub.kiri)
        print(sub.data)
        inord(sub.kanan)
```

```
def postord(sub):
    if sub is not None:
        postord(sub.kiri)
        postord(sub.kanan)
        print(sub.data)

def size(node):
    if node is None:
        return 0
    else:
        return (size(node.kiri)+ 1 + size(node.kanan))

def maxDepth(node):
    if node is None:
        return 0 ;

    else :
        lDepth = maxDepth(node.kiri)
        rDepth = maxDepth(node.kanan)

        if (lDepth > rDepth):
            return lDepth+1
        else:
            return rDepth+1


def traverse(root):
    lvlist=[]
    current_level = [root]
    lv=0
    while current_level:
        #print(' '.join(str(node) for node in current_level))
        next_level = list()
        for n in current_level:
            if n.kiri:
```

2. Hasil Run



3. Membuat program

```
def traverse(root):
    lvlist=[]
    current_level = [root]
    lv=0
    while current_level:
        #print(' '.join(str(node) for node in current_level))
        next_level = list()
        for n in current_level:
            if n.kiri:
                next_level.append(n.kiri)
                level.append(lv+1)
            if n.kanan:
                next_level.append(n.kanan)
                level.append(lv+1)
            current_level = next_level

        lv+=1
        lvlist.append(lv)
    return lvlist

def cetakdatadanlevel(root):
    traverse(A)
    print(root.data, ', Level 0')
    for i in range(len(level)):
          print(datalist[i+1], ', Level', level[i])


print('Ukuran dari Binary Tree adalah', size(A))
print('')
print('Tinggi maksimal dari Binary Tree adalah', maxDepth(A))
print('')
cetakdatadanlevel(A)
```

Hasil Running

```
('Magetan', ', Level 0')
('Ngawi', ', Level', 1)
('Madiun', ', Level', 1)
('Ponorogo', ', Level', 2)
('Solo', ', Level', 2)
('Jombang', ', Level', 2)
('Karanganyar', ', Level', 2)
('Pacitan', ', Level', 3)
('Bojonegoro', ', Level', 3)
>>>
```