Nama   : Ayasha Ninda Maharani

NIM   : L200170143

Kelas   : D

MODUL 3

Collections, Arrays, and Linked Structures

Soal-Soal untuk Mahasiswa

1. Membuat Fungsi
   a. Memastikan matriks konsisten.

```python
a = [[1,2],[3,4]]
b = [[7,2],[1,4]]
c = [[1,"a","b"],[3,4,"c"]]
d = [[2,1],[3,4],[6,5]]
e = [[3,2,1],[5,4,3]]
f = [[1,2,3],[4,5,6],[1,5,6]]

def cekKonsisten(n):
    x = len(n[0])
    z = 0
    for i in range(len(n)):
        if (len(n[i]) == x):
            z+=1
    if(z == len(n)):
        print("Matriks konsisten")
    else:
        print("Matrik tidak konsisten")

cekKonsisten(a)
cekKonsisten(b)
cekKonsisten(c)

def cekInt(n):
    x = 0
    y = 0
    for i in n:
        for j in i:
            y+=1
            if (str(j).isdigit()==False):
                print("Mempunyai Tipe Data yang Berbeda")
                break
            else:
                x+=1
    if(x==y):
        print("Mempunyai Tipe Data yang Sama")

cekInt(a)
cekInt(b)
cekInt(c)
```

Setelah di run :

```
Matriks konsisten
Matriks konsisten
Matriks konsisten
Mempunyai Tipe Data yang Sama
Mempunyai Tipe Data yang Sama
Mempunyai Tipe Data yang Berbeda
Mempunyai Tipe Data yang Berbeda
```

b. Mengambil ukuran matriks.

```python
def ordo(n):
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    print("Mempunyai ordo "+str(x)+"x"+str(y))

ordo(a)
ordo(b)
ordo(d)
ordo(e)
```

Setelah di run :

```
Mempunyai ordo 2x2
Mempunyai ordo 2x2
Mempunyai ordo 3x2
Mempunyai ordo 2x3
```

c. Menjumlahkan matriks.

```python
def jumlah(n,m):
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    xy = [[0 for j in range(x)] for i in range(y)]

    z = 0
    if(len(n)==len(m)):
        for i in range(len(n)):
            if(len(n[i]) == len(m[i])):
                z+=1
    if(z==len(n) and z==len(m)):
        print("Ukuran sama")
        for i in range(len(n)):
            for j in range(len(n[i])):
                xy[i][j] = n[i][j] + m[i][j]
        print(xy)
    else:
        print("Ukuran beda")

jumlah(a,b)
jumlah(a,d)
```

Setelah di run :

```
Ukuran sama
[[8, 4], [4, 8]]
Ukuran beda
```

d. Mengalikan matriks.

```python
def kali(n,m):
    aa = 0
    x,y = 0,0
    for i in range(len(n)):
        x+=1
        y = len(n[i])
    v,w = 0,0
    for i in range(len(m)):
        v+=1
        w = len(m[i])

    if(y==v):
        print("Dapat Dikalikan")
        vwxy = [[0 for j in range(w)] for i in range(x)]
        for i in range(len(n)):
            for j in range(len(m[0])):
                for k in range(len(m)):
                    #print(n[i][k], m[k][j])
                    vwxy[i][j] += n[i][k] * m[k][j]
        print(vwxy)

    else:
        print("Tidak memenuhi syarat")

zz = [[1,2,3],[1,2,3]]
zx = [[1],[2],[3]]
kali(zz,zx)
kali(a,b)
kali(a,e)
kali(a,zx)
```

Setelah di run :

```
Dapat Dikalikan
[[14], [14]]
Dapat Dikalikan
[[9, 10], [25, 22]]
Dapat Dikalikan
[[13, 10, 7], [29, 22, 15]]
Tidak memenuhi syarat
```

e. Menghitung determinan.

```python
def determHitung(A, total=0):
    x = len(A[0])
    z = 0
    for i in range(len(A)):
        if (len(A[i]) == x):
            z+=1
    if(z == len(A)):
        if(x==len(A)):
            indices = list(range(len(A)))
            if len(A) == 2 and len(A[0]) == 2:
                val = A[0][0] * A[1][1] - A[1][0] * A[0][1]
                return val
            for fc in indices:
                As = A
                As = As[1:]
                height = len(As)
                for i in range(height):
                    As[i] = As[i][0:fc] + As[i][fc+1:]
                sign = (-1) ** (fc % 2)
                sub_det = determHitung(As)
                total += sign * A[0][fc] * sub_det
        else:
            return "Tidak bisa dihitung determinan, bukan matrix bujursangkar"
    else:
        return "Tidak bisa dihitung determinan, bukan matrix bujursangkar"
    return total


z = [[4,2],[1,7]]
x = [[3,4,5],[1,3,2],[1,2,3]]
v = [[2,-3,0,0],[2,1,-5,2],[3,1,3,5],[6,7,-8,4]]
r = [[10,22,44,11,12],[2,2,1,1,9],[1,2,3,4,5],[5,2,5,3,8],[1,2,5,3,11]]
print(determHitung(z))
print(determHitung(x))
print(determHitung(v))
print(determHitung(r))
print(determHitung(d))
print(determHitung(e))
```

Setelah di run :

```
26
6
-532
9642
Tidak bisa dihitung determinan, bukan matrix bujursangkar
Tidak bisa dihitung determinan, bukan matrix bujursangkar
>>>
```

2. Membuat matriks dengan *list comprehension*.

a.

```python
def buatNol(n,m=None):
    if(m==None):
        m=n
    print("membuat matriks 0 dengan ordo "+str(n)+"x"+str(m))
    print([[0 for j in range(m)] for i in range(n)])

buatNol(2,4)
buatNol(3)
```

Seterlah di run :

```
membuat matriks 0 dengan ordo 2x4
[[0, 0, 0, 0], [0, 0, 0, 0]]
membuat matriks 0 dengan ordo 3x3
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

b.

```python
def buatIden(n):
    print("membuat matriks identitas dengan ordo"+str(n)+"x"+str(n))
    print([[1 if j==i else 0 for j in range(n)] for i in range(n)])

buatIden(4)
buatIden(2)
```

Setelah di run :

```
membuat matriks identitas dengan ordo4x4
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
membuat matriks identitas dengan ordo2x2
[[1, 0], [0, 1]]
```

3. Membuat fungsi yang terkait dengan *linked list*.

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    def pushAw(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    def pushAk(self, data):
        if (self.head == None):
            self.head = Node(data)
        else:
            current = self.head
            while (current.next != None):
                current = current.next
            current.next = Node(data)
        return self.head

    def insert(self,data,pos):
        node = Node(data)
        if not self.head:
            self.head = node
        elif pos==0:
            node.next = self.head
            self.head = node
        else:
            prev = None
            current = self.head
            current_pos = 0
            while(current_pos < pos) and current.next:
                prev = current
                current = current.next
                current_pos +=1
            prev.next = node
            node.next = current
        return self.head
    def deleteNode(self, position):
        if self.head == None:
            return
        temp = self.head
        if position == 0:
            self.head = temp.next
            temp = None
            return
        for i in range(position -1 ):
            temp = temp.next
            if temp is None:
                break
        if temp is None:
            return
        if temp.next is None:
            return
        next = temp.next.next
        temp.next = None
        temp.next = next
```

```python
    def search(self, x):
        current = self.head
        while current != None:
            if current.data == x:
                return "True"
            current = current.next
        return "False"
    def display(self):
        current = self.head
        while current is not None:
            print(current.data, end ==' ')
            current = current.next

llist = LinkedList()
llist.pushAw(11)
llist.pushAw(32)
llist.pushAw(52)
llist.pushAw(34)
llist.pushAw(3)
llist.pushAw(29)
llist.pushAk(7)
llist.deleteNode(0)
llist.insert(5,1)
print(llist.search(22))
print(llist.search(25))
llist.display()
```

4. Membuat fungsi yang terkait dengan *doubly linked list*.

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
class DoublyLinkedList:
    def __init__(self):
        self.head = None
    def awal(self, new_data):
        print("Menambah simpul pada awal", new_data)
        new_node = Node(new_data)
        new_node.next = self.head
        if self.head is not None:
            self.head.prev = new_node
        self.head = new_node
    def akhir(self, new_data):
        print("Menambah simpul pada akhir", new_data)
        new_node = Node(new_data)
        new_node.next = None
        if self.head is None:
            new_node.prev = None
            self.head = new_node
            return
        last = self.head
        while(last.next is not None):
            last = last.next
        last.next = new_node
        new_node.prev = last
        return

    def printList(self, node):
        print("\nDari Depan :")
        while(node is not None):
            print(" % d" %(node.data))
            last = node
            node = node.next
        print("\nDari Belakang :")
        while(last is not None):
            print(" % d" %(last.data))
            last = last.prev
llist = DoublyLinkedList()
llist.awal(7)
llist.awal(5)
llist.akhir(3)
llist.akhir(2)
llist.printList(llist.head)
```

Setelah di run :

```
Menambah simpul pada awal 7
Menambah simpul pada awal 5
Menambah simpul pada akhir 3
Menambah simpul pada akhir 2

Dari Depan :
   5
   7
   3
   2

Dari Belakang :
   2
   3
   7
   5
```