LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

MODUL 03

"COLLECTIONS, ARRAYS, AND LINKED STRUCTURES"

NAMA : AIZA FRAVY QANZA
NIM : L200170144
KELAS : D
MODUL : 3

Soal-Soal untuk Mahasiswa

1. Array dua dimensi :

Matriks yang akan dites :

```
>>> a = [[1,2],[3,4]]
>>> b = [[7,6],[4,8]]
>>> c = [[12,3,"x","y"],[28,3,99]]
>>> d = [[2,8],[0,3],[9,9]]
>>> e = [[1,8,3],[4,2,6],[7,9,9]]
>>>
```

- Memastikan isi dan ukuran matriks konsisten

```
>>> 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)
matriks konsisten
>>> cekKonsisten(b)
matriks konsisten
>>> cekKonsisten(c)
matrik tidak konsisten
>>> cekKonsisten(d)
matriks konsisten
>>> cekKonsisten(e)
matriks konsisten
```

```
>>> def cekIsiMatriks(n):
        x = 0
        y = 0
        for i in n:
                for j in i:
                        y+=1
                        if (str(j).isdigit()==False):
                                print("Tidak semua isi matriks adalah angka")
                                break
                        else:
                                x+=1
                if(x==y):
                        print("Semua isi matriks adalah angka")


>>> cekIsiMatriks(a)
Semua isi matriks adalah angka
Semua isi matriks adalah angka
>>> cekIsiMatriks(b)
Semua isi matriks adalah angka
Semua isi matriks adalah angka
>>> cekIsiMatriks(c)
Tidak semua isi matriks adalah angka
>>> cekIsiMatriks(d)
Semua isi matriks adalah angka
Semua isi matriks adalah angka
Semua isi matriks adalah angka
>>> cekIsiMatriks(e)
Semua isi matriks adalah angka
Semua isi matriks adalah angka
Semua isi matriks adalah angka
```

- Mengambil ukuran matriks

```
>>> 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)
Mempunyai ordo 2x2
>>> ordo(b)
Mempunyai ordo 2x2
>>> ordo(c)
Mempunyai ordo 2x3
>>> ordo(d)
Mempunyai ordo 3x2
>>> ordo(e)
Mempunyai ordo 3x3
```

- Menjumlahkan 2 matriks

```
>>> def jumlah(n,m):
        p,r = 0,0
        for i in range(len(n)):
                p+=1
                r = len(n[i])
        pr = [[0 for j in range(p)] for i in range(r)]

        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])):
                                        pr[i][j] = n[i][j] + m[i][j]
                        print(pr)
                else:
                        print("Ukuran beda")


>>> jumlah(a,b)
Ukuran sama
[[8, 8], [7, 12]]
>>> jumlah(b,c)
Ukuran beda
```

- Mengalikan 2 matriks

```
>>> 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("Bisa 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")


>>> kali(a,b)
Bisa dikalikan
[[7, 0], [0, 0]]
[[15, 0], [0, 0]]
[[15, 6], [0, 0]]
[[15, 22], [0, 0]]
[[15, 22], [21, 0]]
[[15, 22], [37, 0]]
[[15, 22], [37, 18]]
[[15, 22], [37, 50]]
>>> kali(d,e)
Tidak memenuhi syarat
```

- Menghitung determinan matriks

```
>>> def detHitung(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 = detHitung(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

>>> detHitung(a)
-2
>>> detHitung(b)
32
>>> detHitung(c)
'tidak bisa dihitung determinan, bukan matrix bujursangkar'
>>> detHitung(d)
'tidak bisa dihitung determinan, bukan matrix bujursangkar'
>>> detHitung(e)
78
```

2. Matriks dan list comprehension
   - Membangkitkan matriks berisi nol semua

```
>>> 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(1,3)
Membuat matriks 0 dengan ordo 1x3
[[0, 0, 0]]
>>> buatNol(2)
Membuat matriks 0 dengan ordo 2x2
[[0, 0], [0, 0]]
```

   - Membangkitkan matriks identitas

```
>>> def buatIdentitas(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)])


>>> buatIdentitas(3)
membuat matriks identitas dengan ordo3x3
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
>>> buatIdentitas(2)
membuat matriks identitas dengan ordo2x2
[[1, 0], [0, 1]]
```

3. Linked list
    - Mencari data yang isinya tertentu
    - Menambah suatu simpul diawal
    - Menambah suatu simpul diakhir
    - Menyisipkan suatu simpul dimana saja
    - Mengahapus suatu simpul dimana saja

```
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
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
```

```
>>> ll=LinkedList()
>>> ll.pushAw(28)
>>> ll.pushAw(3)
>>> ll.pushAw(19)
>>> ll.pushAw(99)
>>> ll.pushAw(10)
>>> ll.pushAk(3)
<__main__.Node object at 0x0000014C6771EA90>
>>> print(ll.display())
10 99 19 3 28 3 None
>>> ll.deleteNode(0)
>>> print(ll.display())
99 19 3 28 3 None
>>> ll.insert(12,5)
<__main__.Node object at 0x0000014C6770BA58>
>>> print(ll.display())
99 19 3 28 12 3 None
>>> print(ll.search(28))
True
>>> print(ll.search(12))
True
>>> print(ll.search(70))
False
```

4. Doubly Linked List
    - Mengunjungi dan mencetak data tiap simpul dari depan dan dari belakang
    - Menambah suatu simpul diawal
    - Menambah suatu simpul diakhir

```
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 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 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
```

```
>>> lli = DoublyLinkedList()
>>> lli.awal(3)
Menambah pada awal 3
>>> lli.awal(28)
Menambah pada awal 28
>>> Menambah pada akhir 19
SyntaxError: invalid syntax
>>> lli.akhir(19)
Menambah pada akhir 19
>>> lli.akhir(99)
Menambah pada akhir 99
>>> lli.printList(lli.head)

Dari Depan :
  28
  3
  19
  99

Dari Belakang :
  99
  19
  3
  28
```