

Nama : Riska Meilina Puspa

NIM : L200180192

Kelas : G

Modul 2

Praktikum Algoritma dan Struktur Data

1.

```
a = [[1,2], [3,4]]
b = [[5,6], [7,8]]
c = [[12,3,"y"], [12,33,4]]
d = [[3,4], [2,4], [1,5]]
e = [[5,6,7], [7,8,9]]
f = [[2,3], [4,5,6], [7,8,9]]

def cekKonsis(n):
    x = len(n[0])
    y = type(n[0][0])
    z = 0
    a = True
    for i in range(len(n)):
        for j in range(len(n[i])):
            #mengecek apakah matriks mempunyai isi yg bertipe sama
            c = type(n[i][j])
            if (c!=y):
                a = False
                break
        #mengecek apakah matriks mempunyai ukuran yg sama
        if (len(n[i]) == x):
            z+=1

    if(z == len(n) and a==True):
        print("matriks konsisten")
    else:
        print("matrik tidak konsisten")

cekKonsis(a)
cekKonsis(f)
cekKonsis(c)
```

```

def cekInt(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")

```

```

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

```

```

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

```

```

ordo(a)
ordo(b)
ordo(d)
ordo(f)

```

```

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)

```

```

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)]
        print(vwxy)
        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)

```

```

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 = [[3,1],[2,5]]
x = [[1,2,1],[3,3,1],[2,1,2]]
v = [[1,-2,0,0],
      [3,2,-3,1],
      [4,0,5,1],
      [2,3,-1,4]]
r = [[10,23,45,12,13],
      [1,2,3,4,5],
      [1,2,3,4,6],
      [4,2,3,4,8],
      [1,4,5,6,10]]
print(determHitung(z))
print(determHitung(x))
print(determHitung(v))
print(determHitung(r))
print(determHitung(d))
print(determHitung(e))

```

```

matriks konsisten
matrik tidak konsisten
matrik tidak konsisten
semua isi matriks adalah angka
semua isi matriks adalah angka
tidak semua isi matriks adalah angka
2
mempunyai ordo 2x2
2
mempunyai ordo 2x2
3
mempunyai ordo 3x2
3
mempunyai ordo 3x3
ukuran sama
[[6, 8], [10, 12]]
ukuran beda
bisa dikalikan
[[0], [0]]
[[14], [14]]
bisa dikalikan
[[0, 0], [0, 0]]
[[19, 22], [43, 50]]
bisa dikalikan
[[0, 0, 0], [0, 0, 0]]
[[19, 22, 25], [43, 50, 57]]
tidak memenuhi syarat
13
-6
200
330
tidak bisa dihitung determinan, bukan matrix bujursangkar
tidak bisa dihitung determinan, bukan matrix bujursangkar

```

2.

```

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)

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)

```

```

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]]
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.

```

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
            node.next = prev.next
            prev.next = node
        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 ):
            prev = temp
            temp = temp.next
            if temp is None:
                break
        if temp is None:
            return
        if temp.next is None:
            return
        prev.next = temp.next
        temp= None

    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

```

```

l1list = LinkedList()
l1list.pushAw(21)
l1list.pushAw(22)
l1list.pushAw(12)
l1list.pushAw(14)
l1list.pushAw(2)
l1list.pushAw(19)
l1list.pushAk(9)
l1list.display()
l1list.deleteNode(5)
l1list.insert(1,5)
print(l1list.search(21))
print(l1list.search(29))
l1list.display()

```

```

19 2 14 12 22 21 9 False
False
19 2 14 12 22 1 9

```

4.

```
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

l1list = DoublyLinkedList()
l1list.awal(7)
l1list.awal(1)
l1list.akhir(6)
l1list.akhir(4)
l1list.printList(l1list.head)
```


menambah pada awal 7
menambah pada awal 1
menambah pada akhir 6
menambah pada akhir 4

Dari Depan :

1
7
6
4

Dari Belakang :

4
6
7
1