Nama : Windiapriani Ginayawati

NIM : L200170157

Kelas : D

Modul : III

1.

```
print("No. 1")
a = [[3,5],[5,3]]
b = [[4,7],[7,4]]
c = [[1,2,"p","y"],[5,9,0]]
d = [[6,9],[8,2],[6,0]]
e = [[0,2,1],[4,0,7]]
f = [[1,6,0],[3,2,0],[1,9,2]]

def cek(k):
    l = len(k[0])
    count = 0
    for i in range(len(k)):
        if (len(k[i]) == l):
            count+=1
    if(count == len(k)):
        print("Matriks-nya konsisten.")
    else:
        print("Matrik-nya tidak konsisten.")

cek(a)
cek(b)
cek(c)
cek(d)
cek(e)
cek(f)

def cekIsi(k):
    x = 0
    y = 0
    for i in k:
        for j in i:
            y+=1
            if (str(j).isdigit()==False):
                print("Tidak semua isi matriks berisi angka.")
                break
            else:
                x+=1
    if(x==y):
        print("Semua isi matriks berisi angka.")

cekIsi(a)
cekIsi(b)
cekIsi(c)
```

```
def ordoMatriks(k):
    x,y = 0,0
    for i in range(len(k)):
        x+=1
        y = len(k[i])
    print("Matriks mempunyai ordo "+str(x)+"x"+str(y))

ordoMatriks(a)
ordoMatriks(b)
ordoMatriks(d)
ordoMatriks(e)

def penjumlahan(k,l):
    x,y = 0,0
    for i in range(len(k)):
        x+=1
        y = len(k[i])
    xy = [[0 for j in range(x)] for i in range(y)]

    z = 0
    if(len(k)==len(l)):
        for i in range(len(k)):
            if(len(k[i]) == len(l[i])):
                z+=1
    if(z==len(k) and z==len(l)):
        print("Ukuran matriks sama.")
        for i in range(len(k)):
            for j in range(len(k[i])):
                xy[i][j] = k[i][j] + l[i][j]
        print(xy)
    else:
        print("Ukuran matriks berbeda.")

penjumlahan(a,b)
penjumlahan(a,d)
```

```
def perkalian(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.")

bb = [[1,2,3],[1,1,4]]
cc = [[4],[5],[6]]
perkalian(bb,cc)
perkalian(a,b)
perkalian(a,e)
perkalian(a,cc)
```

```
def hitungDet(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 = hitungDet(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 = [[1,2],[4,3]]
x = [[-3,4,5],[3,3,2],[1,2,3]]
v = [[2,-3,0,0],[1,4,1,2],[4,8,6,9],[1,4,-8,4]]
r = [[16,24,32,8,12],[5,4,3,2,1],[1,2,3,4,5],[3,7,0,1,4],[1,1,5,21,11]]
print(hitungDet(z))
print(hitungDet(x))
print(hitungDet(v))
print(hitungDet(r))
print(hitungDet(d))
print(hitungDet(e))
```

Hasil dari No. 1

```
No. 1
Matriks-nya konsisten.
Matriks-nya konsisten.
Matrik-nya tidak konsisten.
Matriks-nya konsisten.
Matriks-nya konsisten.
Matriks-nya konsisten.
Semua isi matriks berisi angka.
Semua isi matriks berisi angka.
Tidak semua isi matriks berisi angka.
Matriks mempunyai ordo 2x2
Matriks mempunyai ordo 2x2
Matriks mempunyai ordo 3x2
Matriks mempunyai ordo 2x3
Ukuran matriks sama.
[[7, 12], [12, 7]]
Ukuran matriks berbeda.
Bisa dikalikan.
[[32], [33]]
Bisa dikalikan.
[[47, 41], [41, 47]]
Bisa dikalikan.
[[20, 6, 38], [12, 10, 26]]
Tidak memenuhi syarat.
-5
-28
463
25824
Tidak bisa dihitung determinan, bukan matrix bujursangkar.
Tidak bisa dihitung determinan, bukan matrix bujursangkar.
```

2.
```
print("No. 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)
buatNol(4,2)

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(4)
buatIdentitas(3)
buatIdentitas(2)
```

Hasil No. 2

```
No. 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 0 dengan ordo 4x2
[[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 ordo3x3
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
Membuat matriks identitas dengan ordo2x2
[[1, 0], [0, 1]]
>>>
```

3.

```
print("No. 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
            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
llist = LinkedList()
llist.pushAw(14)
llist.pushAw(28)
llist.pushAw(61)
llist.pushAw(15)
llist.pushAw(5)
llist.pushAw(25)
llist.pushAk(7)
llist.deleteNode(0)
llist.insert(3,2)
print(llist.search(7))
print(llist.search(29))
llist.display()
```

Hasil no. 3

```
No. 3
True
False
5 15 3 61 28 14 7
```

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
llist = DoublyLinkedList()
llist.awal(7)
llist.awal(3)
llist.akhir(1)
llist.akhir(4)
llist.printList(llist.head)
```

Hasil no. 4

```
No. 4
menambah pada awal 7
menambah pada awal 3
menambah pada akhir 1
menambah pada akhir 4

Dari Depan :
  3
  7
  1
  4

Dari Belakang :
  4
  1
  7
  3
```