ALGORITHM AND DATA STRUCTURE PRACTICUM

MODULE 3

COLEETIONS, ARRA AND LINKED STRUKTURES

CREATED BY :

KURNIAWAN BAGASKARA

L200214253

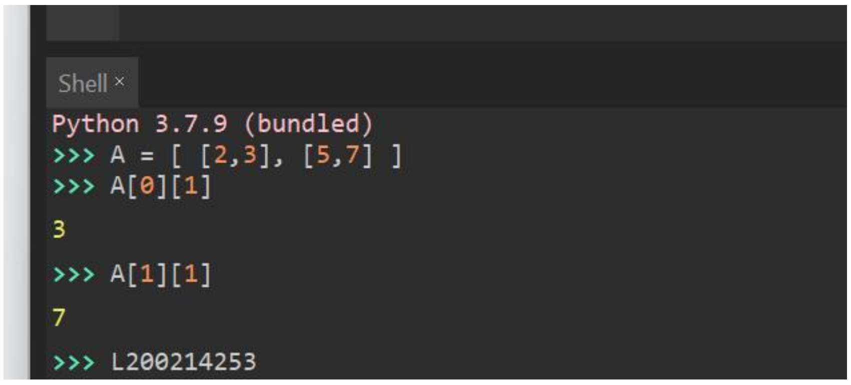
INFORMATICS STUDY PROGRAM

FACULTY OF COMMUNICATION AND INFORMATION SCIENCE

MUHAMMADIYAH SURAKARTA UNIVERSITY

Latihan 3.1

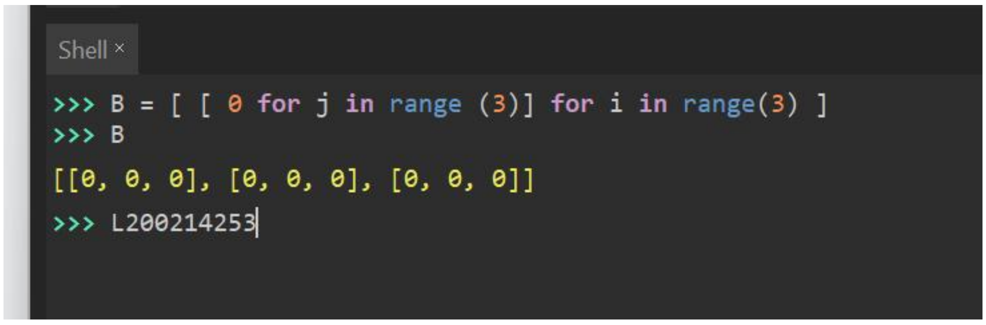
```
>>> A = [ [2,3], [5,7] ]
>>> A[0][1]
3
>>> A[1][1]
7
```



```
Shell ×
Python 3.7.9 (bundled)
>>> A = [ [2,3], [5,7] ]
>>> A[0][1]
3
>>> A[1][1]
7
>>> L200214253
```

Latihan 3.2

```
>>> B = [ [ 0 for j in range (3)] for i in range(3) ]
>>> B
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> L200214253
```



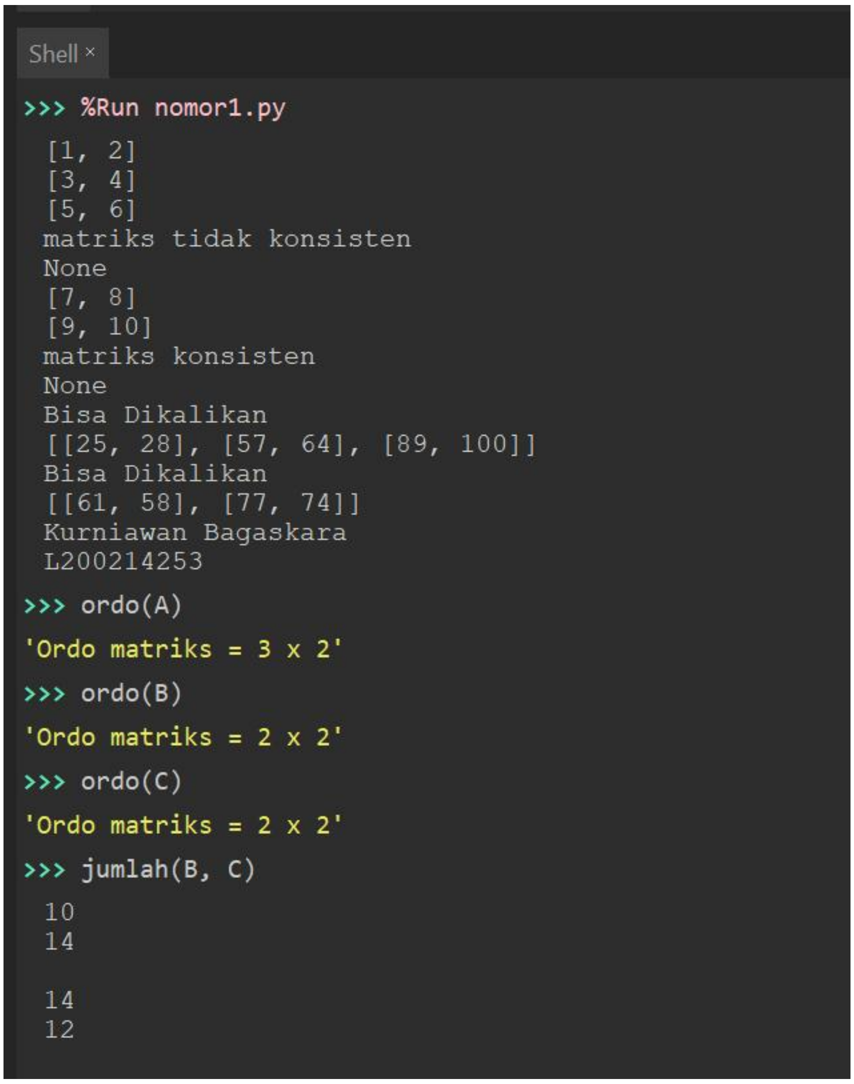
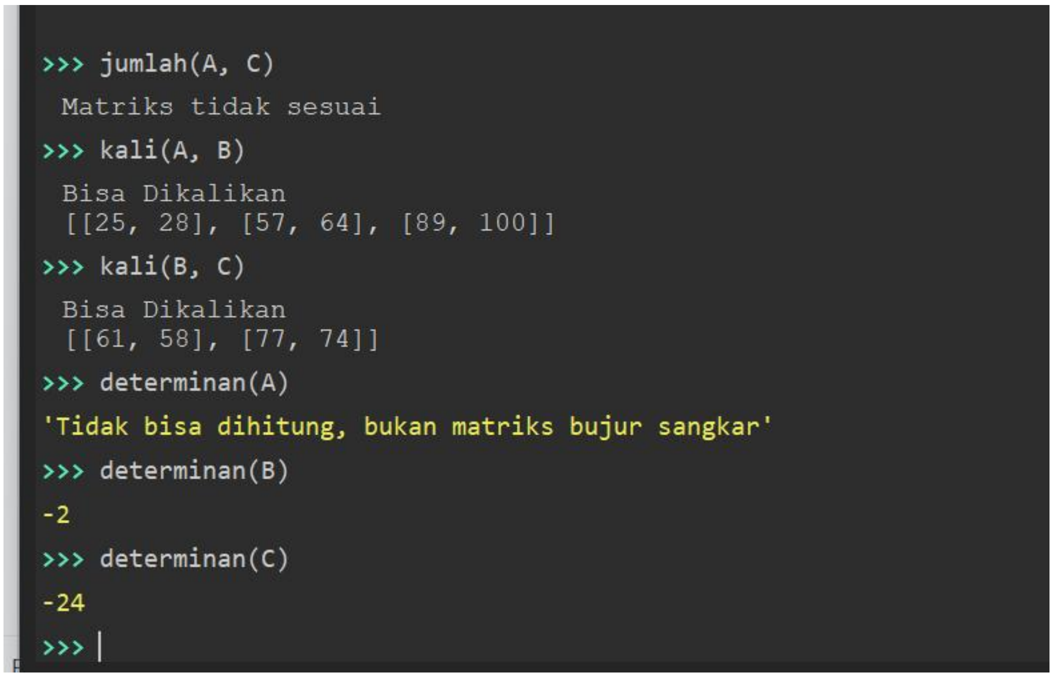
```
Shell ×
>>> B = [ [ 0 for j in range (3)] for i in range(3) ]
>>> B
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> L200214253
```

Soal 1.

A = [[1,2],[3,4],[5,6]]

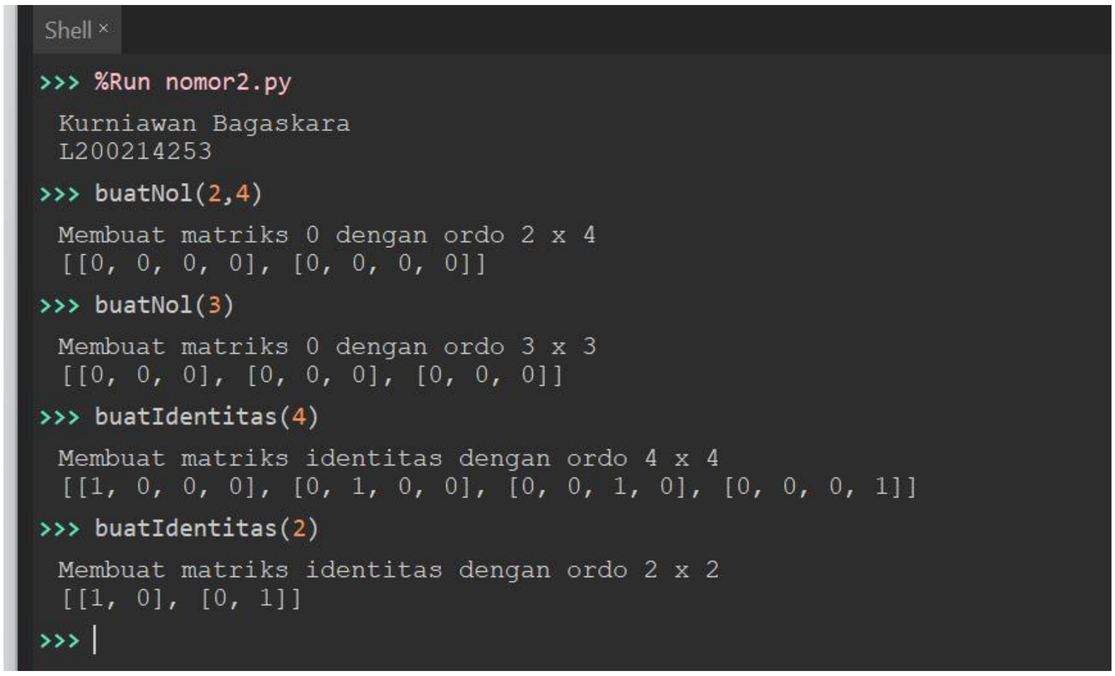
B = [[7,8],[9,10]]

```
C = [[3,6],[5,2]]

#Nomor 1A
class matriks (object):
    def cetakmatriks(self, matriks):
        for i in matriks:
            print(i)
    def cekkonsisten(self, matriks):
        if len(matriks[0]) == len(matriks):
            print ("matriks konsisten")
        else:
            print ("matriks tidak konsisten")

x = matriks()
x.cetakmatriks(A)
print(x.cekkonsisten(A))

y = matriks()
y.cetakmatriks(B)
print(y.cekkonsisten(B))

#Nomor 1B
def ordo(matriks):
    return ("Ordo matriks = "+str(len(matriks))+" x "+str(len(matriks[0])))

#Nomor 1C
def jumlah(matriks1, matriks2):
    if ordo(matriks1) == ordo(matriks2):
        for x in range(0, len(matriks1)):
            for y in range(0, len(matriks1[0])):
                print (matriks1[x][y] + matriks2[x][y],' '),
            print()
    else:
        print("Matriks tidak sesuai")

#Nomor 1D
def kali(m,n):
    a = 0
    x,y = 0,0
    for i in range(len(m)):
        x += 1
        y = len(m[i])
    v,w = 0,0
    for i in range(len(n)):
        v += 1
        w = len(n[i])

    if (y == v):
        print ("Bisa Dikalikan")
        vwxy = [[0 for j in range(w)] for i in range(x)]
        for i in range(len(m)):
            for j in range(len(n[0])):
                for k in range(len(n)):
                    vwxy[i][j] += m[i][k] * n[k][j]
        print (vwxy)
    else:
        print("Tidak memenuhi syarat")

kali(A,B)
kali(B,C)

#Nomor 1E
def determinan(p, total = 0):
    x = len(p[0])
    z = 0
    for i in range(len(p)):
        if (len(p[i]) == x):
            z += 1
    if (z == len(p)):
        if (x == len(p)):
            indices = list(range(len(p)))
            if len(p) == 2 and len(p[0]) == 2:
                val = p[0][0] * p[1][1] - p[1][0] * p[0][1]
                return val
            for fc in indices:
                pq = p
                pq = pq[1:]
                height = len(pq)
                for i in range(height):
                    pq[i] = pq[i][0:fc] + pq[i][fc+1:]
                sign = (-1) ** (fc % 2)
                sub_det = determinanHitung(pq)
                total += sign * A[0][fc] * sub_det
        else:
            return "Tidak bisa dihitung, bukan matriks bujur sangkar"
    else:
        return "Tidak bisa dihitung, bukan matriks bujur sangkar"
    return total
print('Kurniawan Bagaskara')
print('L200214253')
```


```
Shell ×
>>> %Run nomor1.py
 [1, 2]
 [3, 4]
 [5, 6]
 matriks tidak konsisten
 None
 [7, 8]
 [9, 10]
 matriks konsisten
 None
 Bisa Dikalikan
 [[25, 28], [57, 64], [89, 100]]
 Bisa Dikalikan
 [[61, 58], [77, 74]]
 Kurniawan Bagaskara
 L200214253
>>> ordo(A)
'Ordo matriks = 3 x 2'
>>> ordo(B)
'Ordo matriks = 2 x 2'
>>> ordo(C)
'Ordo matriks = 2 x 2'
>>> jumlah(B, C)
 10
 14

 14
 12
```

```
>>> jumlah(A, C)
 Matriks tidak sesuai
>>> kali(A, B)
 Bisa Dikalikan
 [[25, 28], [57, 64], [89, 100]]
>>> kali(B, C)
 Bisa Dikalikan
 [[61, 58], [77, 74]]
>>> determinan(A)
'Tidak bisa dihitung, bukan matriks bujur sangkar'
>>> determinan(B)
-2
>>> determinan(C)
-24
>>> 
```

Soal 2.

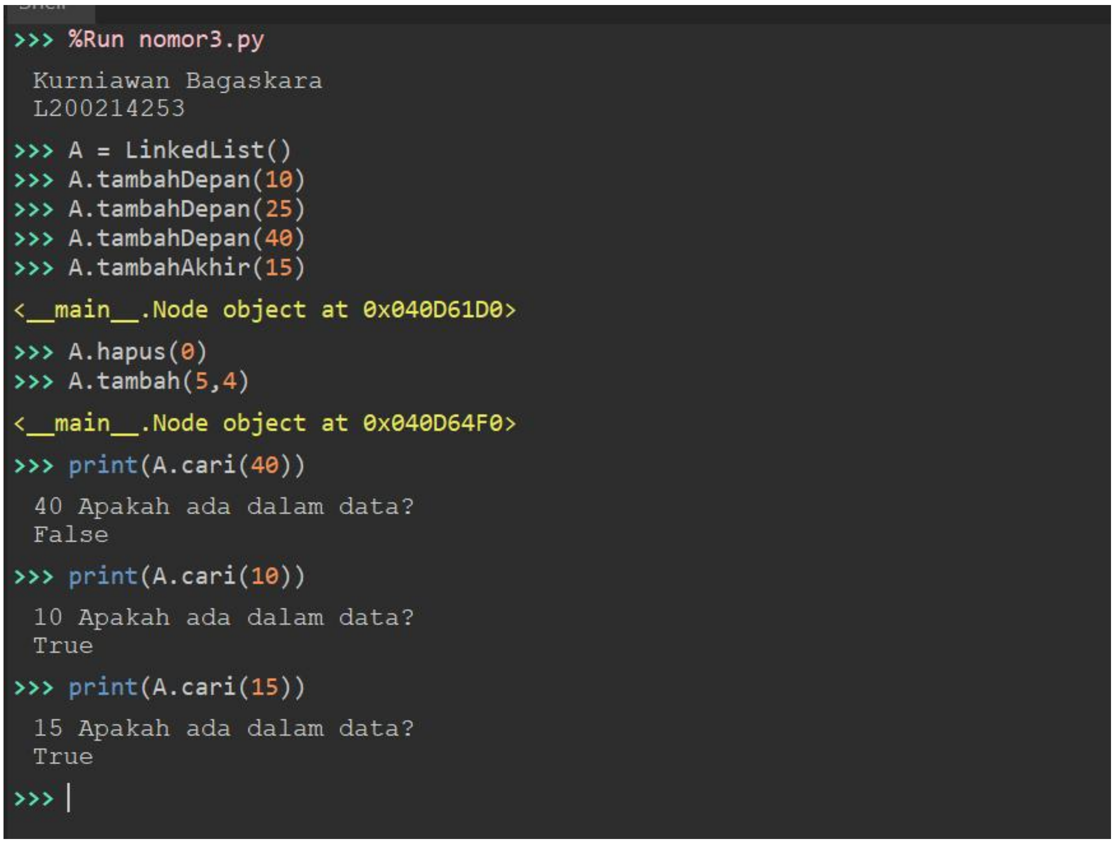
```
#Nomor 2A
def buatNol(n, m=None):
    if (m == None):
        m = n
    print ("Membuat matriks 0 dengan ordo "+str(n)+" x "+str(m))
    print ([[0 for j in range(m)] for i in range(n)])

#Nomor 2B
def buatIdentitas(m):
    n = m
    print("Membuat matriks identitas dengan ordo "+str(n)+" x "+str(n))
    matriks = [[1 if j == i else 0 for j in range(m)] for i in range(n)]
    print(matriks)
print('Kurniawan Bagaskara')
print('L200214253')
```



```
>>> %Run nomor2.py
  Kurniawan Bagaskara
  L200214253
>>> buatNol(2,4)
  Membuat matriks 0 dengan ordo 2 x 4
  [[0, 0, 0, 0], [0, 0, 0, 0]]
>>> buatNol(3)
  Membuat matriks 0 dengan ordo 3 x 3
  [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> buatIdentitas(4)
  Membuat matriks identitas dengan ordo 4 x 4
  [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
>>> buatIdentitas(2)
  Membuat matriks identitas dengan ordo 2 x 2
  [[1, 0], [0, 1]]
>>> |
```

Soal 3.

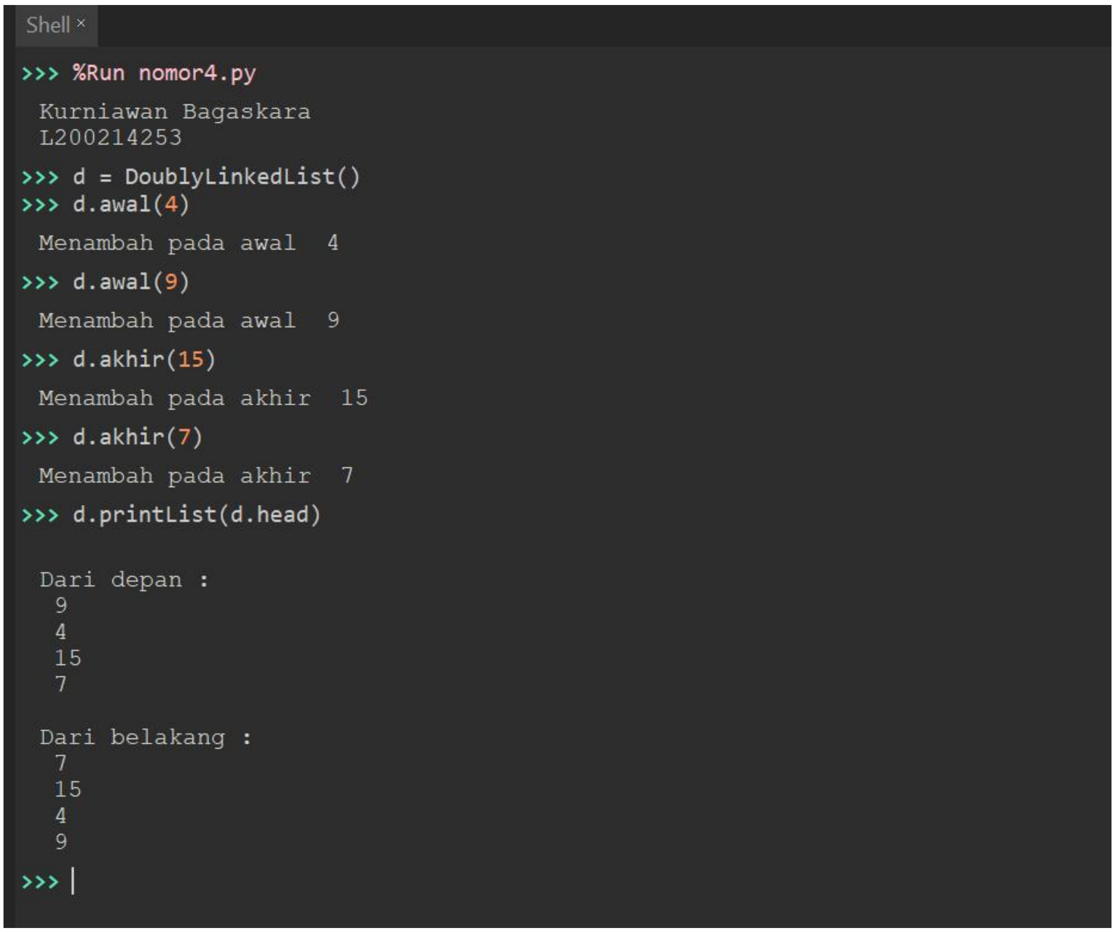
```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.head = None
    def tambahDepan(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    def tambahAkhir(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 tambah(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 hapus(self,posisi):
        if self.head == None:
            return
        temp = self.head
        if posisi == 0:
            self.head = temp.next
            temp = None
            return
        for i in range(posisi - 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 cari(self,x):
        current = self.head
        while current != None:
            if current.data == x:
                print(x, "Apakah ada dalam data?")
                return True
            current = current.next
        print(x,"Apakah ada dalam data?")
        return False
    def display(self):
        current = self.head
        while current is not None:
            print(current.data, end = ' ')
            current = current.next
print('Kurniawan Bagaskara')
print('L200214253')
```



```
>>> %Run nomor3.py
  Kurniawan Bagaskara
  L200214253
>>> A = LinkedList()
>>> A.tambahDepan(10)
>>> A.tambahDepan(25)
>>> A.tambahDepan(40)
>>> A.tambahAkhir(15)
<__main__.Node object at 0x040D61D0>
>>> A.hapus(0)
>>> A.tambah(5,4)
<__main__.Node object at 0x040D64F0>
>>> print(A.cari(40))
  40 Apakah ada dalam data?
  False
>>> print(A.cari(10))
  10 Apakah ada dalam data?
  True
>>> print(A.cari(15))
  15 Apakah ada dalam data?
  True
>>> |
```

Soal 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
print('Kurniawan Bagaskara')
print('L200214253')
```


Shell ×

```
>>> %Run nomor4.py
 Kurniawan Bagaskara
 L200214253
>>> d = DoublyLinkedList()
>>> d.awal(4)
 Menambah pada awal  4
>>> d.awal(9)
 Menambah pada awal  9
>>> d.akhir(15)
 Menambah pada akhir  15
>>> d.akhir(7)
 Menambah pada akhir  7
>>> d.printList(d.head)

 Dari depan :
  9
  4
  15
  7

 Dari belakang :
  7
  15
  4
  9
>>> 
```