LATIHAN DAN TUGAS PRAKTIKUM

MODUL 3

PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

DISUSUN OLEH:

Nama : Syifaul Qolbi Auliya' Darojat

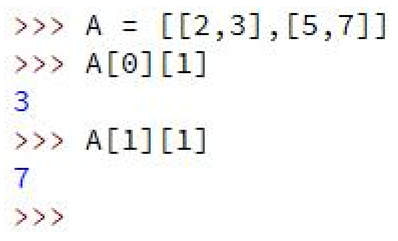
NIM : L200200141

KELAS : F

UNIVERSITAS MUHAMMADIYAH SURAKARTA

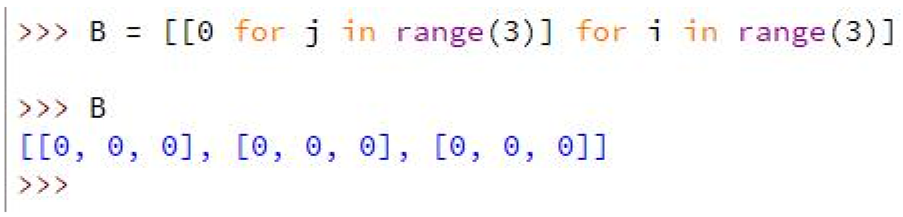
LATIHAN

Latihan 1



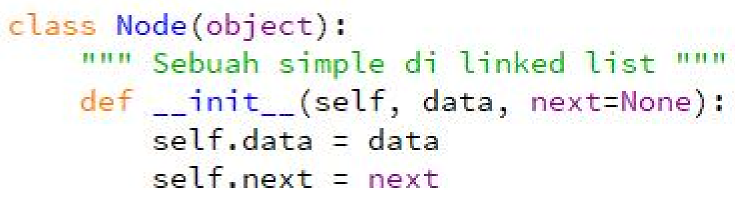
```
>>> A = [[2,3],[5,7]]
>>> A[0][1]
3
>>> A[1][1]
7
>>>
```

Latihan 2



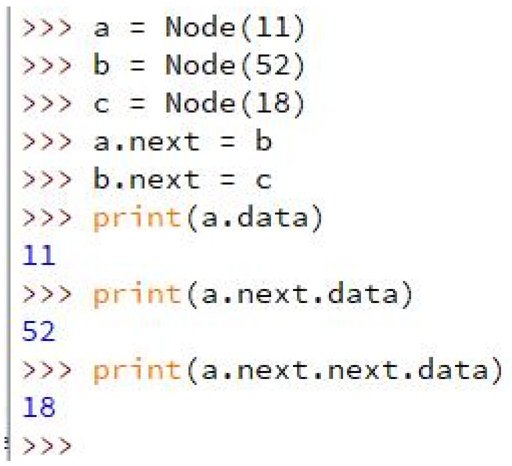
```
>>> B = [[0 for j in range(3)] for i in range(3)]

>>> B
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
```

Linked List



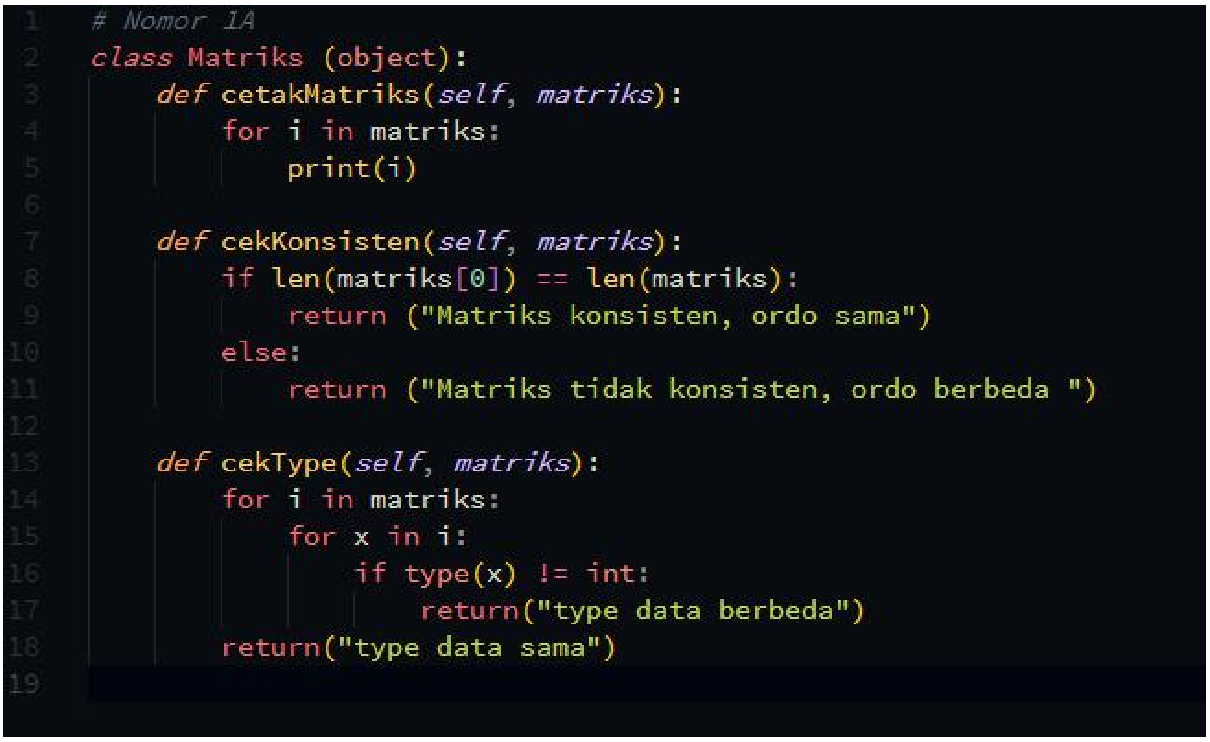
```
class Node(object):
    """ Sebuah simple di linked list """
    def __init__(self, data, next=None):
        self.data = data
        self.next = next
```

Hasil :



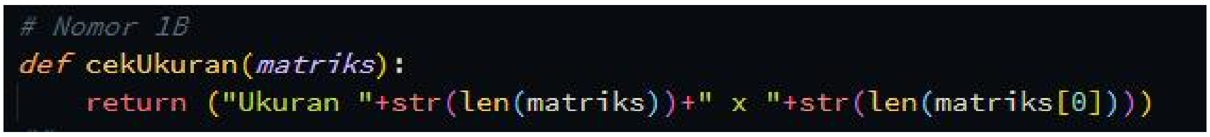
```
>>> a = Node(11)
>>> b = Node(52)
>>> c = Node(18)
>>> a.next = b
>>> b.next = c
>>> print(a.data)
11
>>> print(a.next.data)
52
>>> print(a.next.next.data)
18
>>>
```

1A



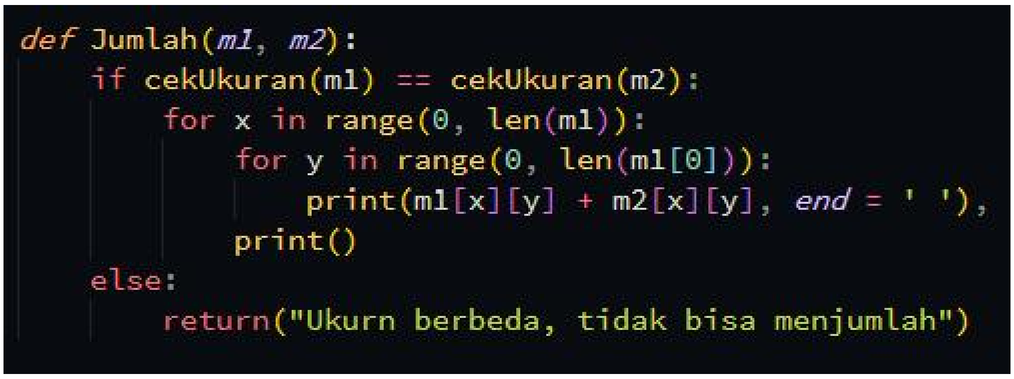
```python
# Nomor 1A
class Matriks (object):
    def cetakMatriks(self, matriks):
        for i in matriks:
            print(i)

    def cekKonsisten(self, matriks):
        if len(matriks[0]) == len(matriks):
            return ("Matriks konsisten, ordo sama")
        else:
            return ("Matriks tidak konsisten, ordo berbeda ")

    def cekType(self, matriks):
        for i in matriks:
            for x in i:
                if type(x) != int:
                    return("type data berbeda")
        return("type data sama")
```

1B



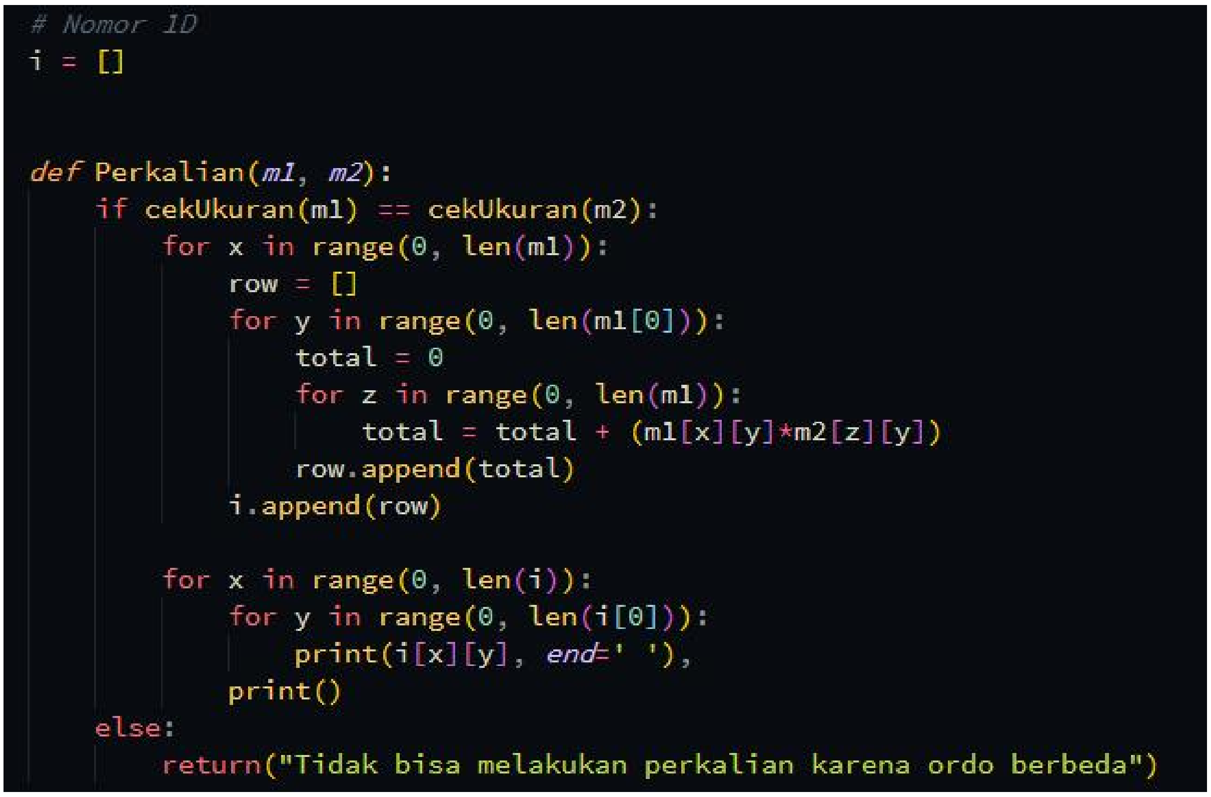
```python
# Nomor 1B
def cekUkuran(matriks):
    return ("Ukuran "+str(len(matriks))+" x "+str(len(matriks[0])))
```

1C

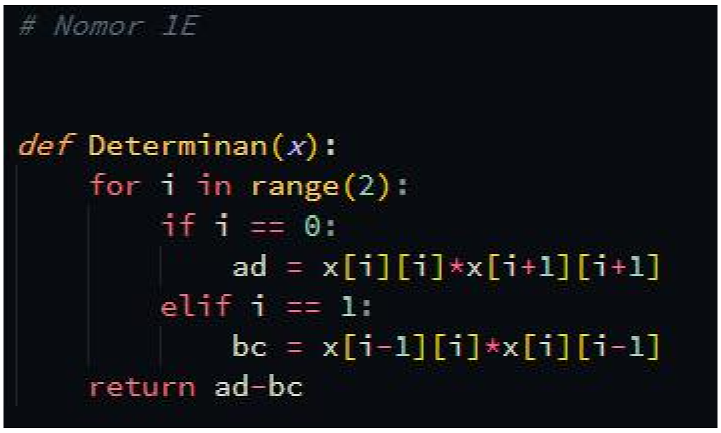


```python
def Jumlah(m1, m2):
    if cekUkuran(m1) == cekUkuran(m2):
        for x in range(0, len(m1)):
            for y in range(0, len(m1[0])):
                print(m1[x][y] + m2[x][y], end = ' '),
            print()
    else:
        return("Ukurn berbeda, tidak bisa menjumlah")
```

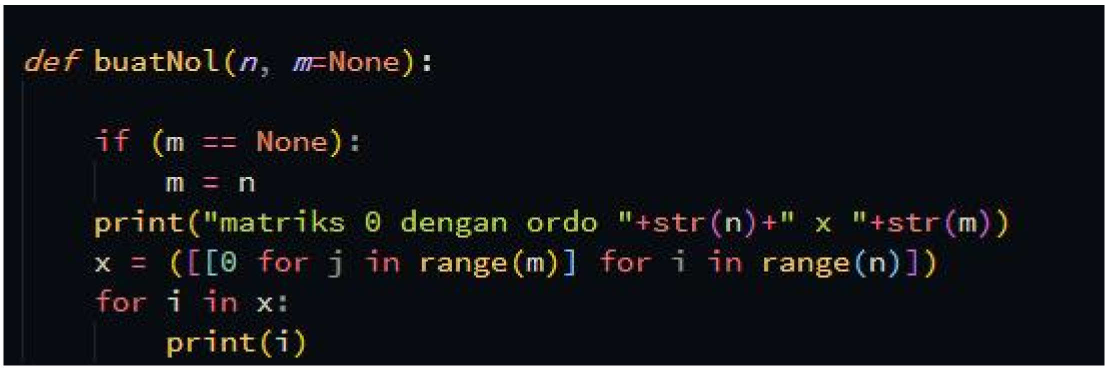
1D


```python
# Nomor 1D
i = []


def Perkalian(m1, m2):
    if cekUkuran(m1) == cekUkuran(m2):
        for x in range(0, len(m1)):
            row = []
            for y in range(0, len(m1[0])):
                total = 0
                for z in range(0, len(m1)):
                    total = total + (m1[x][y]*m2[z][y])
                row.append(total)
            i.append(row)

        for x in range(0, len(i)):
            for y in range(0, len(i[0])):
                print(i[x][y], end=' '),
            print()
    else:
        return("Tidak bisa melakukan perkalian karena ordo berbeda")
```

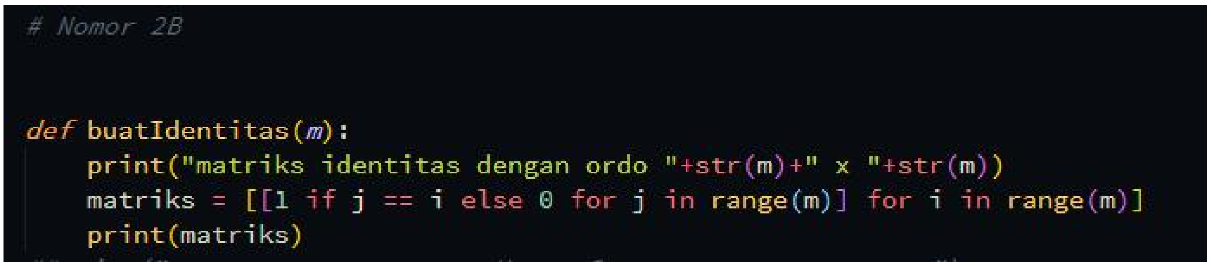
1E


```python
# Nomor 1E


def Determinan(x):
    for i in range(2):
        if i == 0:
            ad = x[i][i]*x[i+1][i+1]
        elif i == 1:
            bc = x[i-1][i]*x[i][i-1]
    return ad-bc
```

2A

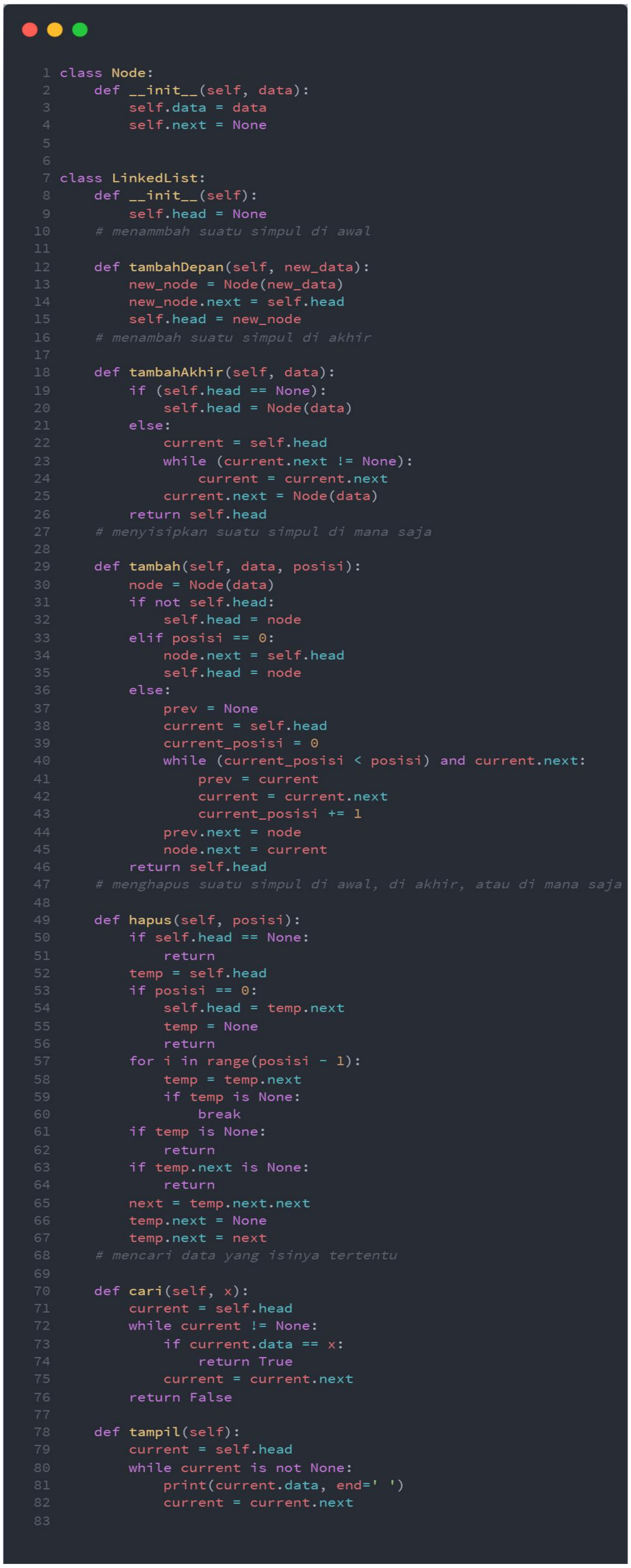

```python
def buatNol(n, m=None):

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

2B



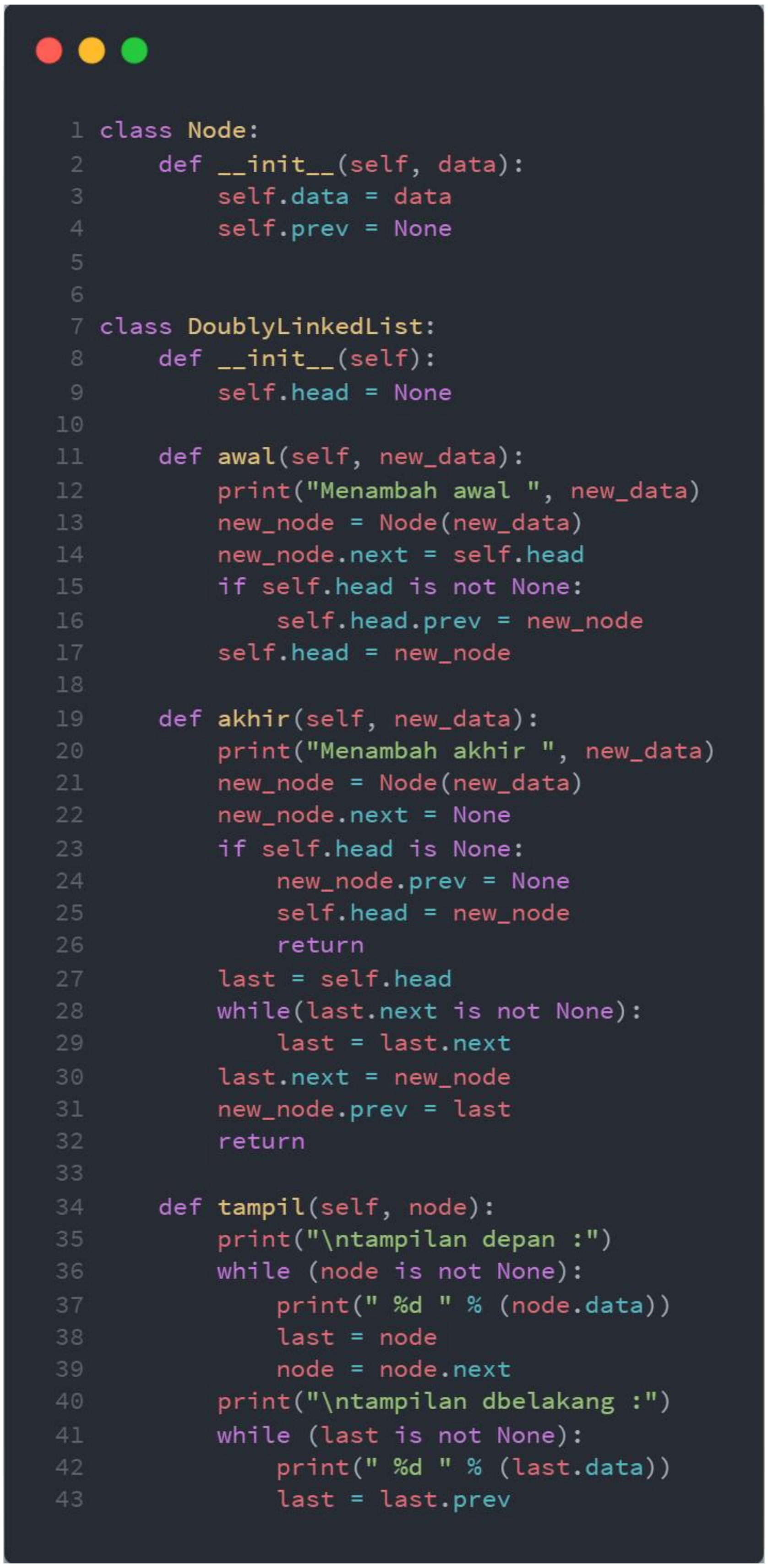
```python
# Nomor 2B


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

3



```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    def __init__(self):
        self.head = None
    # menammbah suatu simpul di awal

    def tambahDepan(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    # menambah suatu simpul di akhir

    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
    # menyisipkan suatu simpul di mana saja

    def tambah(self, data, posisi):
        node = Node(data)
        if not self.head:
            self.head = node
        elif posisi == 0:
            node.next = self.head
            self.head = node
        else:
            prev = None
            current = self.head
            current_posisi = 0
            while (current_posisi < posisi) and current.next:
                prev = current
                current = current.next
                current_posisi += 1
            prev.next = node
            node.next = current
        return self.head
    # menghapus suatu simpul di awal, di akhir, atau di mana saja

    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
    # mencari data yang isinya tertentu

    def cari(self, x):
        current = self.head
        while current != None:
            if current.data == x:
                return True
            current = current.next
        return False

    def tampil(self):
        current = self.head
        while current is not None:
            print(current.data, end=' ')
            current = current.next
```

4



```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 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 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 tampil(self, node):
        print("\ntampilan depan :")
        while (node is not None):
            print(" %d " % (node.data))
            last = node
            node = node.next
        print("\ntampilan dbelakang :")
        while (last is not None):
            print(" %d " % (last.data))
            last = last.prev
```