PRAKTIKUM ALGORITMA STRUKTUR DATA

MODUL 3

COLLECTIONS, ARRAYS, AND LINKED STRUCTURES

Disusun oleh:

Adinda Aulia Hapsari

L200220037

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

TAHUN 2024

Setelah kegiatan selesai, lembar kerja ini dicetak (di-print) dan dikumpulkan ke asisten. NIM : L200220037 Nama : Adinda Aulia Hapsari Nama Asisten : Tanggal Praktikum :	(Diisi oleh Asisten) Nilai Praktek : Tanda Tangan :

1. Terkait array dua dimensi, kita akan membuat tipe data sebuah matrix yang berisi angka-angka. Untuk itu buatlah fungsi-fungsi:
   a. Untuk memastikan bahwa isi dan ukuran matrix-nya konsisten.
   b. Untuk mengambil ukuran matrixnya.
   c. Untuk menjumlahkan dua matrix.
   d. Untuk mengalikan dua matrix.
   e. Untuk menghitung determinan sebuah matrix.

```
print('\n--- Oleh L200220037 ---')

class Matrix:
    #point 1
    def __init__(self, data):
        self.data = data
        self.rows = len(data)
        self.cols = len(data[0])
        self.validasi()

    def validasi(self):
        for baris in self.data:
            if len(baris) != self.cols:
                raise ValueError("Matriks memiliki panjang baris yang tidak
konsisten.")
            for elemen in baris:
                if not isinstance(elemen, (int, float)):
                    raise ValueError("Elemen-elemen matriks harus berupa
bilangan bulat atau pecahan.")
#point 2
    def ukuran(self):
        return self.rows, self.cols
#point 3
    def penjumlahan(self, lainnya):
        if self.ukuran() != lainnya.ukuran():
            raise ValueError("Matriks harus memiliki ukuran yang sama.")
        hasil = []
        for i in range(self.rows):
            hasil.append([self.data[i][j] + lainnya.data[i][j] for j in
range(self.cols)])
        return Matrix(hasil)
```

```
#point 4
    def perkalian(self, lainnya):
        if self.cols != lainnya.rows:
            raise ValueError("Jumlah kolom matriks pertama harus sama
dengan jumlah baris matriks kedua.")
        hasil = []
        for i in range(self.rows):
            baris_hasil = []
            for j in range(lainnya.cols):
                elemen_hasil = sum(self.data[i][k] * lainnya.data[k][j] for
k in range(self.cols))
                baris_hasil.append(elemen_hasil)
            hasil.append(baris_hasil)
        return Matrix(hasil)
#point 5
    def hitungDeterminan(self):
        if self.rows != self.cols:
            raise ValueError("Determinan hanya dapat dihitung untuk matriks
persegi.")
        if self.rows == 1:
            return self.data[0][0]
        elif self.rows == 2:
            return self.data[0][0] * self.data[1][1] - self.data[0][1] *
self.data[1][0]
        else:
            det = 0
            for i in range(self.rows):
                det += (-1) ** i * self.data[0][i] * self.submatriks(0,
i).hitungDeterminan()
            return det

    def submatriks(self, baris, kolom):
        return Matrix([baris[:kolom] + baris[kolom + 1:] for baris in
(self.data[:baris] + self.data[baris + 1:])])


matrix1 = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix2 = Matrix([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

print("Matrix 1:")
print(matrix1.data)
print("Matrix 2:")
print(matrix2.data)

print("Ukuran Matrix 1:", matrix1.ukuran())
print("Ukuran Matrix 2:", matrix2.ukuran())

print("Penjumlahan Matrix:")
print(matrix1.penjumlahan(matrix2).data)

print("Perkalian Matrix:")
```

```
print(matrix1.perkalian(matrix2).data)

print("Determinan Matrix 1:", matrix1.hitungDeterminan())
```

```
--- Oleh L200220037 ---
Matrix 1:
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Matrix 2:
[[9, 8, 7], [6, 5, 4], [3, 2, 1]]
Ukuran Matrix 1: (3, 3)
Ukuran Matrix 2: (3, 3)
Penjumlahan Matrix:
[[10, 10, 10], [10, 10, 10], [10, 10, 10]]
Perkalian Matrix:
[[30, 24, 18], [84, 69, 54], [138, 114, 90]]
Determinan Matrix 1: 0
```

2. Terkait matrix dan list compherension, buatlah fungsi-fungsi:
   a. Untuk membangkitkan matrix berisi nol semua, dengan diberikan ukurannya.
   b. Untuk membangkitkan matrix identitas, dengan diberikan ukurannya.

```
print('\n--- Oleh L200220037 ---')

class Matrix:
    def __init__(self, data=None):
        if data is None:
            data = []
        self.data = data

    @classmethod
    def buatNol(cls, m, n=None):
        if n is None:
            n = m
        return cls([[0] * n for _ in range(m)])

    @classmethod
    def buatIdentitas(cls, m):
        return cls([[1 if i == j else 0 for j in range(m)] for i in
range(m)])


# Contoh Penggunaan
matriks_nol_3x3 = Matrix.buatNol(3, 3)
matriks_nol_4x4 = Matrix.buatNol(4)
matriks_identitas_3x3 = Matrix.buatIdentitas(3)

print("Matriks Nol 3x3:")
print(matriks_nol_3x3.data)

print("\nMatriks Nol 4x4:")
```

```
print(matriks_nol_4x4.data)

print("\nMatriks Identitas 3x3:")
print(matriks_identitas_3x3.data)
```

```
--- Oleh L200220037 ---
Matriks Nol 3x3:
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

Matriks Nol 4x4:
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

Matriks Identitas 3x3:
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
```

3. Terkait linked list, buatlah fungsi untuk:
   a. Mencari data yang isinya tertentu.
   b. Menambah suatu simpul di awal.
   c. Menambah suatu simpul di akhir.
   d. Menyisipkan suatu simpul dimana saja.
   e. Menghapus simpul di awal, di akhir, atau di mana saja.

```
print('\n--- Oleh L200220037 ---')

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, 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 cari(self, x):
        current = self.head
        while current != None:
            if current.data == x:
                return "True"
            current = current.next
        return "False"
    def tampilkan(self):
        current = self.head
        while current is not None:
            print(current.data, end = ' ')
            current = current.next

llist = LinkedList()
llist.tambahDepan(21)
llist.tambahDepan(22)
llist.tambahDepan(12)
llist.tambahDepan(14)
llist.tambahDepan(2)
llist.tambahDepan(19)
llist.tambahAkhir(9)
```

```
llist.hapus(0)
llist.tambah(1,6)
print(llist.cari(21))
print(llist.cari(29))
llist.tampilkan()
```

```
--- Oleh L200220037 ---
True
False
2 14 12 22 21 1 9
```

4. Terkait doubly linked list, buatlah fungsi untuk:
   a. Mengunjungi dan mencetak data tiap simpul dari depan dan dari belakang.
   b. Menambah suatu simpul di awal.
   c. Menambah suatu simpul di akhir.

```
print('\n--- Oleh L200220037 ---')

class Simpul:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None

class DaftarBerantaiGanda:
    def __init__(self):
        self.head = None

#Membuat simpul awal (poin2)
    def tambah_di_awal(self, data):
        # Membuat simpul baru
        new_node = Simpul(data)

        # Jika daftar kosong
        if self.head is None:
            self.head = new_node
        else:
            new_node.next = self.head
            self.head.prev = new_node
            self.head = new_node

#Membuat simpul akhir (poin 3)
    def tambah_di_akhir(self, data):
        # Membuat simpul baru
        new_node = Simpul(data)

        # Jika daftar kosong
        if self.head is None:
```

```
            self.head = new_node
        else:
            current = self.head
            # Menemukan simpul terakhir
            while current.next:
                current = current.next
            # Menambahkan simpul baru setelah simpul terakhir
            current.next = new_node
            new_node.prev = current

#(poin 1)
    def cetak_maju(self):
        current = self.head
        while current:
            print(current.data, end=" ")
            current = current.next
        print()

    def cetak_mundur(self):
        current = self.head
        while current and current.next:
            current = current.next
        while current:
            print(current.data, end=" ")
            current = current.prev
        print()

# Contoh penggunaan
daftar = DaftarBerantaiGanda()
daftar.tambah_di_awal(1)
daftar.tambah_di_awal(2)
daftar.tambah_di_awal(3)
daftar.tambah_di_akhir(4)
daftar.tambah_di_akhir(5)

print("Doubly Linked List dari depan:")
daftar.cetak_maju()

print("Doubly Linked List dari belakang:")
daftar.cetak_mundur()
```

```
--- Oleh L200220037 ---
Doubly Linked List dari depan:
3 2 1 4 5
Doubly Linked List dari belakang:
5 4 1 2 3
```