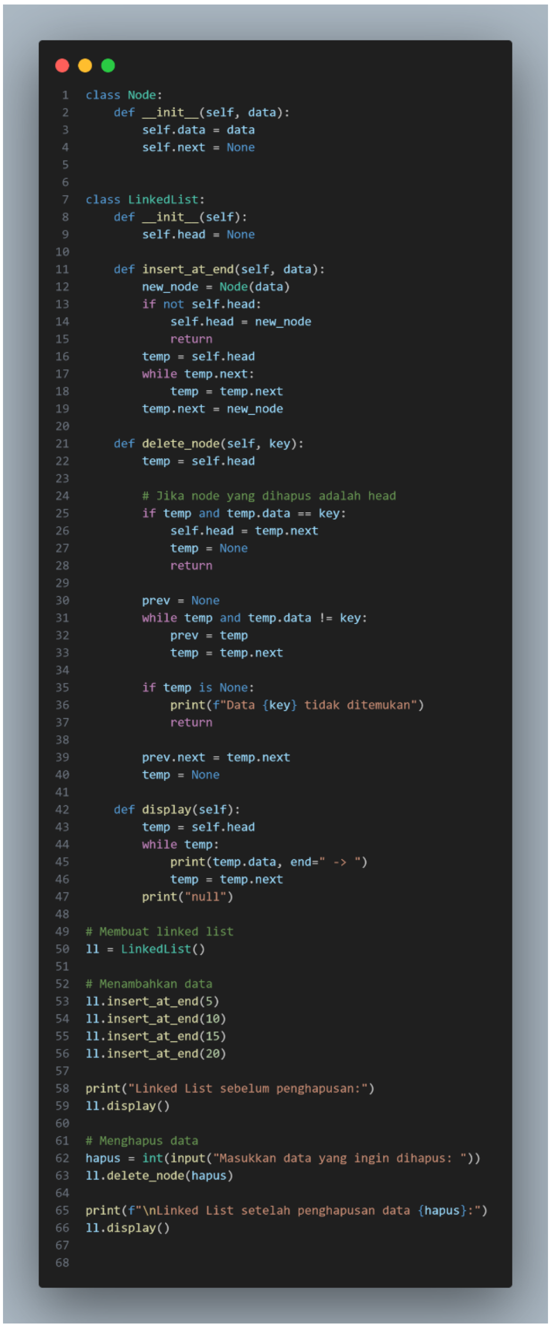
Pertemuan 3

Rafif Muhammad Faiz

J0403251024

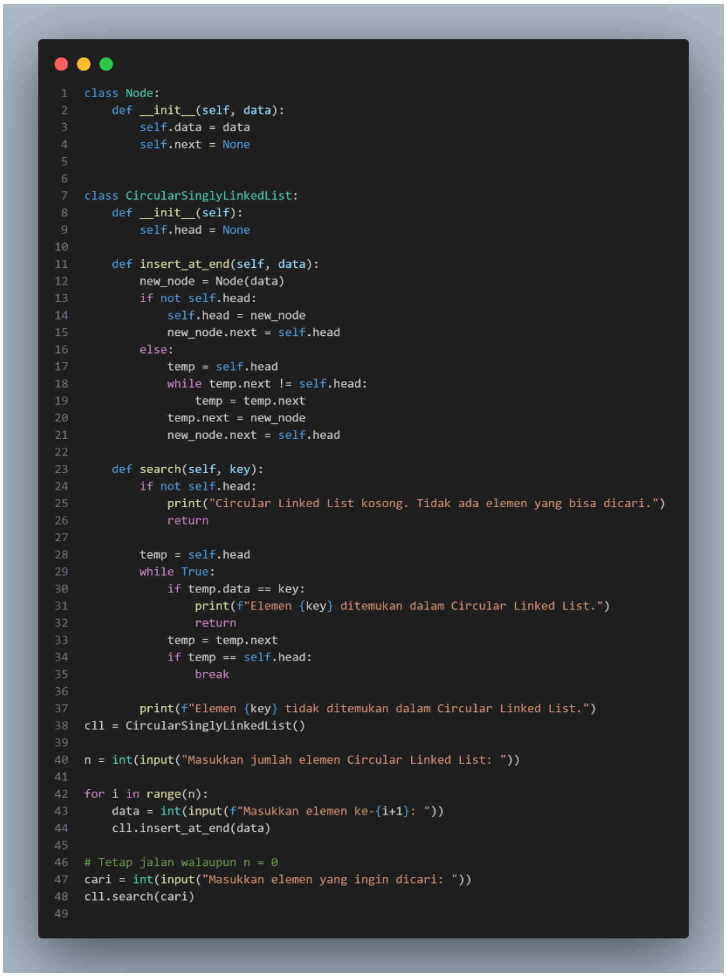
TPL B/ P2

1. Tugas 1



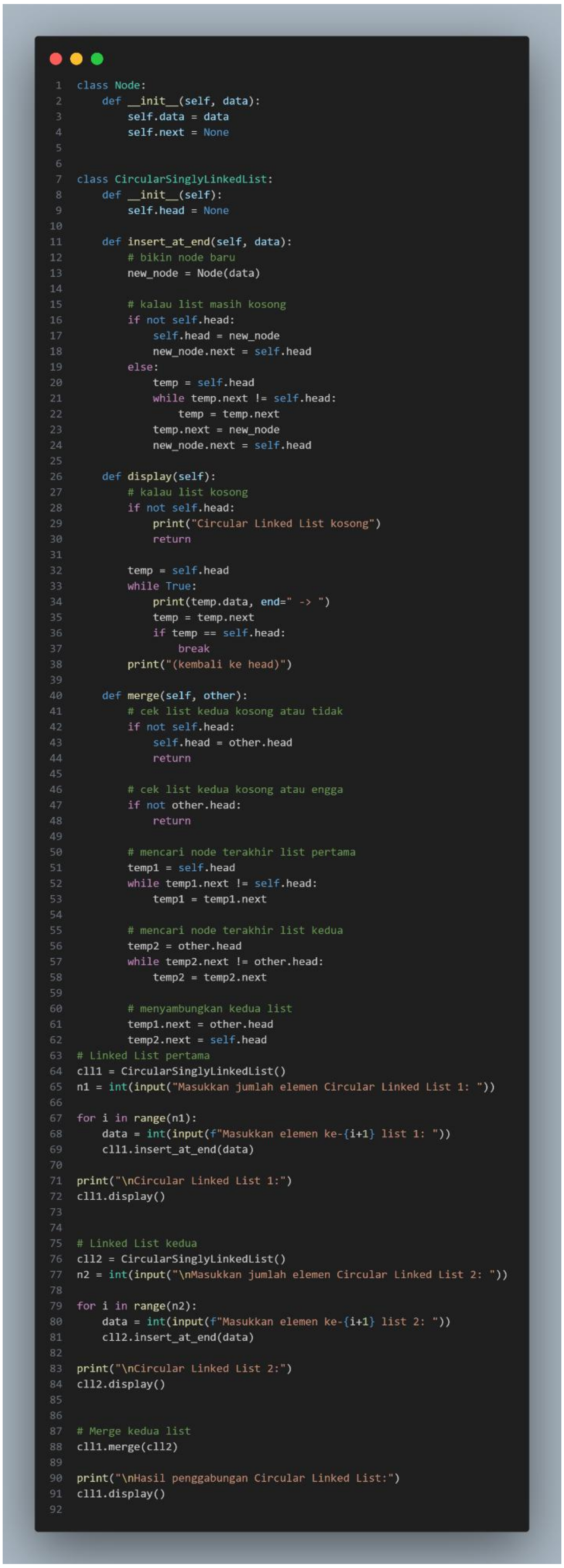
```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


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

    def insert_at_end(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        temp = self.head
        while temp.next:
            temp = temp.next
        temp.next = new_node

    def delete_node(self, key):
        temp = self.head

        # Jika node yang dihapus adalah head
        if temp and temp.data == key:
            self.head = temp.next
            temp = None
            return

        prev = None
        while temp and temp.data != key:
            prev = temp
            temp = temp.next

        if temp is None:
            print(f"Data {key} tidak ditemukan")
            return

        prev.next = temp.next
        temp = None

    def display(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("null")

# Membuat linked list
ll = LinkedList()

# Menambahkan data
ll.insert_at_end(5)
ll.insert_at_end(10)
ll.insert_at_end(15)
ll.insert_at_end(20)

print("Linked List sebelum penghapusan:")
ll.display()

# Menghapus data
hapus = int(input("Masukkan data yang ingin dihapus: "))
ll.delete_node(hapus)

print(f"\nLinked List setelah penghapusan data {hapus}:")
ll.display()


```

2. Tugas 2



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


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

    def insert_at_end(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            new_node.next = self.head
        else:
            temp = self.head
            while temp.next != self.head:
                temp = temp.next
            temp.next = new_node
            new_node.next = self.head

    def search(self, key):
        if not self.head:
            print("Circular Linked List kosong. Tidak ada elemen yang bisa dicari.")
            return

        temp = self.head
        while True:
            if temp.data == key:
                print(f"Elemen {key} ditemukan dalam Circular Linked List.")
                return
            temp = temp.next
            if temp == self.head:
                break

        print(f"Elemen {key} tidak ditemukan dalam Circular Linked List.")
cll = CircularSinglyLinkedList()

n = int(input("Masukkan jumlah elemen Circular Linked List: "))

for i in range(n):
    data = int(input(f"Masukkan elemen ke-{i+1}: "))
    cll.insert_at_end(data)

# Tetap jalan walaupun n = 0
cari = int(input("Masukkan elemen yang ingin dicari: "))
cll.search(cari)

```

3. Tugas 4



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


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

    def insert_at_end(self, data):
        # bikin node baru
        new_node = Node(data)

        # kalau list masih kosong
        if not self.head:
            self.head = new_node
            new_node.next = self.head
        else:
            temp = self.head
            while temp.next != self.head:
                temp = temp.next
            temp.next = new_node
            new_node.next = self.head

    def display(self):
        # kalau list kosong
        if not self.head:
            print("Circular Linked List kosong")
            return

        temp = self.head
        while True:
            print(temp.data, end=" -> ")
            temp = temp.next
            if temp == self.head:
                break
        print("(kembali ke head)")

    def merge(self, other):
        # cek list kedua kosong atau tidak
        if not self.head:
            self.head = other.head
            return

        # cek list kedua kosong atau engga
        if not other.head:
            return

        # mencari node terakhir list pertama
        temp1 = self.head
        while temp1.next != self.head:
            temp1 = temp1.next

        # mencari node terakhir list kedua
        temp2 = other.head
        while temp2.next != other.head:
            temp2 = temp2.next

        # menyambungkan kedua list
        temp1.next = other.head
        temp2.next = self.head
# Linked List pertama
cll1 = CircularSinglyLinkedList()
n1 = int(input("Masukkan jumlah elemen Circular Linked List 1: "))

for i in range(n1):
    data = int(input(f"Masukkan elemen ke-{i+1} list 1: "))
    cll1.insert_at_end(data)

print("\nCircular Linked List 1:")
cll1.display()


# Linked List kedua
cll2 = CircularSinglyLinkedList()
n2 = int(input("\nMasukkan jumlah elemen Circular Linked List 2: "))

for i in range(n2):
    data = int(input(f"Masukkan elemen ke-{i+1} list 2: "))
    cll2.insert_at_end(data)

print("\nCircular Linked List 2:")
cll2.display()


# Merge kedua list
cll1.merge(cll2)

print("\nHasil penggabungan Circular Linked List:")
cll1.display()

```