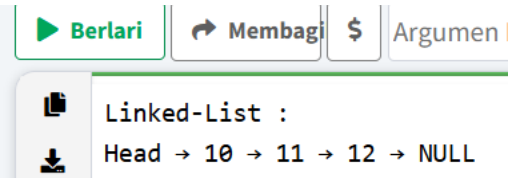
Praktek 22

```
# function untuk membuat node
def buat_node(data):
    return {'data': data, 'next': None}

# menambahkan node di akhir list
def tambah_node(head, data):
    new_node = buat_node(data)
    if head is None:
        return new_node
    current = head
    while current['next'] is not None:
        current = current['next']
    current['next'] = new_node
    return head

# menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")
```

```
# Contoh Penerapan
# Head awal dari linked-list
head = None

# Tambah node
head = tambah_node(head, 10)
head = tambah_node(head, 11)
head = tambah_node(head, 12)

# cetak linked-list
print('Linked-List : ')
cetak_linked_list(head)
```

Hasil Run :



Penjelasannya :

Baris 1 Mendefinisikan fungsi buat_node(data) untuk membuat node baru. Node berupa dictionary dengan dua key: 'data' untuk menyimpan nilai dan 'next' untuk menunjuk ke node berikutnya (default-nya None).

Baris 2 Fungsi tambah_node(head, data) mulai didefinisikan. Fungsinya untuk menambahkan node di akhir linked list.

Baris 3 Membuat node baru dengan memanggil fungsi buat_node(data) dan menyimpannya ke variabel new_node.

Baris 4 Jika head bernilai None, berarti linked list masih kosong → node baru langsung dikembalikan sebagai head.

Baris 5 -7 Jika tidak kosong, program mencari node terakhir (yang next-nya None) dengan perulangan.

Baris 8 Node baru ditambahkan di akhir list dengan cara mengatur next dari node terakhir agar menunjuk ke new_node.

Baris 9 Mengembalikan head sebagai kepala linked list yang telah diperbarui.

Baris 11 Mendefinisikan fungsi cetak_linked_list(head) untuk mencetak isi linked list dari awal hingga akhir.

Baris 12 Membuat variabel current yang akan dipakai untuk iterasi mulai dari head.

Baris 13 Mencetak kata “Head →” sebagai penanda awal linked list.

Baris 14 - 16 Selama current tidak None, cetak data node lalu lanjut ke node berikutnya.

Baris 17 Mencetak NULL untuk menunjukkan akhir linked list.

Baris 20 Inisialisasi variabel head dengan None sebagai linked list kosong.

Baris 22 – 24 Menambahkan tiga node ke linked list masing-masing dengan data 10, 11, dan 12.

Baris 26 Menampilkan teks "Linked-List :" sebelum mencetak isi linked list.

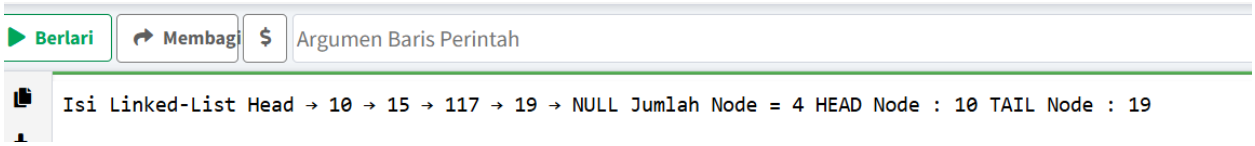
Baris 27 Memanggil fungsi cetak_linked_list(head) untuk mencetak seluruh isi linked list.

Praktek 23 :

```
# function untuk membuat node
def buat_node(data):
    return {'data': data, 'next': None}

# menambahkan node di akhir list
def tambah_node(head, data):
    new_node = buat_node(data)
    if head is None:
        return new_node
    current = head
    while current['next'] is not None:
        current = current['next']
    current['next'] = new_node
    return head

# traversal untuk cetak isi linked-list
def traversal_to_display(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")
```

```
# traversal untuk menghitung jumlah elemen dalam linked-list
def traversal_to_count_nodes(head):
    count = 0
    current = head
    while current is not None:
        count += 1
        current = current['next']
    return count

# traversal untuk mencari dimana tail (node terakhir)
def traversal_to_get_tail(head):
    if head is None:
        return None
    current = head
    while current['next'] is not None:
        current = current['next']
    return current


# cetak isi linked-list
print("Isi Linked-List")
traversal_to_display(head)

# cetak jumlah node
print("Jumlah Nodes = ", traversal_to_count_nodes(head))

# cetak HEAD node
print("HEAD Node : ", head['data'])

# cetak TAIL NODE
print("TAIL Node : ", traversal_to_get_tail(head)['data'])
```

Hasil Run :



Penjelasannya :

Baris 1 Komentar yang menjelaskan bahwa fungsi di bawah digunakan untuk membuat node baru.

Baris 2 Mendefinisikan fungsi buat_node yang menerima satu parameter data.

Baris 3 Fungsi mengembalikan sebuah node berbentuk dictionary yang terdiri dari dua bagian: data (berisi nilai) dan next (penunjuk ke node berikutnya, awalnya None).

Baris 5 Komentar yang menjelaskan bahwa fungsi berikut digunakan untuk menambahkan node di akhir linked list.

Baris 6 Mendefinisikan fungsi tambah_node dengan dua parameter: head sebagai kepala list, dan data sebagai nilai untuk node baru.

Baris 7 Membuat node baru menggunakan fungsi buat_node dan menyimpannya ke variabel lokal.

Baris 8 Memeriksa apakah linked list masih kosong.

Baris 9 Jika linked list kosong, maka node baru dijadikan sebagai head.

Baris 10 Jika list tidak kosong, traversal dimulai dari head.

Baris 11 Perulangan dilakukan untuk mencari node terakhir (yang next-nya masih ada).

Baris 12 Pointer berpindah ke node berikutnya selama masih ada node.

Baris 13 Setelah node terakhir ditemukan, node baru disambungkan ke akhir list melalui atribut next.

Baris 14 Mengembalikan head sebagai hasil akhir.

Baris 16 Komentar yang menunjukkan bahwa fungsi selanjutnya digunakan untuk mencetak isi dari linked list.

Baris 17 Mendefinisikan fungsi traversal_to_display yang menerima head sebagai parameter.

Baris 18 Pointer current mulai diarahkan ke head.

Baris 19 Menampilkan teks "Head" untuk menunjukkan awal dari linked list.

Baris 20 Perulangan dimulai selama masih ada node dalam list.

Baris 21 Setiap kali iterasi, data dari node saat ini dicetak.

Baris 22 Pointer berpindah ke node berikutnya.

Baris 23 Setelah semua node dicetak, ditampilkan teks NULL sebagai penanda akhir list.

Baris 25 Komentar yang menjelaskan bahwa fungsi berikut digunakan untuk menghitung jumlah node.

Baris 26 Mendefinisikan fungsi traversal_to_count_nodes dengan parameter head.

Baris 27 Menginisialisasi variabel penghitung jumlah node.

Baris 28 Pointer current mulai dari head.

Baris 29 Perulangan berjalan selama masih ada node.

Baris 30 Setiap kali melewati node, penghitung ditambah satu.

Baris 31 Pointer pindah ke node berikutnya.

Baris 32 Jumlah total node dikembalikan.

Baris 34 Komentar bahwa fungsi berikut digunakan untuk mencari node terakhir (tail) dalam linked list.

Baris 35 Mendefinisikan fungsi traversal_to_get_tail yang menerima head.

Baris 36 Jika linked list kosong, fungsi langsung mengembalikan None.

Baris 37 Pointer dimulai dari head.

Baris 38 Perulangan dilakukan selama masih ada node berikutnya.

Baris 39 Pointer terus berpindah ke node berikutnya.

Baris 40 Jika node terakhir ditemukan (yang next-nya None), node tersebut dikembalikan.

Baris 43 Komentar yang menyatakan bahwa bagian berikut merupakan bagian implementasi (penggunaan) fungsi.

Baris 44 Linked list diinisialisasi sebagai kosong.

Baris 45–48
Menambahkan empat node dengan nilai berbeda ke dalam linked list, satu per satu di akhir list.

Baris 50 Komentar bahwa bagian ini akan mencetak isi linked list.

Baris 51 Menampilkan teks untuk memberi tahu bahwa isi linked list akan ditampilkan.

Baris 52 Memanggil fungsi pencetak linked list untuk menampilkan seluruh isi list dari awal hingga akhir.

Baris 54 Komentar bahwa bagian ini digunakan untuk menghitung dan menampilkan jumlah node.

Baris 55 Mencetak jumlah node dalam list dengan memanggil fungsi penghitung.

Baris 57 Komentar bahwa bagian ini digunakan untuk mencetak node pertama (head).

Baris 58 Menampilkan data dari node pertama (head).

Baris 60 Komentar bahwa bagian ini digunakan untuk mencetak node terakhir (tail).

Baris 61 Menampilkan data dari node terakhir menggunakan fungsi pencari tail.

Praktek 24 :

```
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan membuat linked-list awal
head = None
head = sisip_depan(head, 30)
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penyisipan di Depan")
cetak = cetak_linked_list(head)
```

```
# Penyisipan node
data = 99
head = sisip_depan(head, data)

print("\nData Yang Disispkan : ", data)

# cetak isi setelah penyisipan node baru di awal
print("\nIsi Linked-List Setelah Penyisipan di Depan")
cetak_linked_list(head)
```

Run :

```
Isi Linked-List Sebelum Penyisipan di Depan Head → 10 → 20 → 30 → NULL Data Yang Disispkan : 99 Isi
Linked-List Setelah Penyisipan di Depan Head → 99 → 10 → 20 → 30 → NULL
```

Penjelasannya :

Baris 1: Menandai bahwa fungsi berikutnya digunakan untuk membuat node baru.
Baris 2: Mendefinisikan fungsi sisip_depan dengan dua parameter: head dan data.
Baris 3: Membuat node baru dalam bentuk dictionary dengan elemen 'data' dan 'next'. 'data' berisi nilai yang akan disimpan, sedangkan 'next' menunjuk ke node sebelumnya (head saat ini).
Baris 4: Mengembalikan node baru yang dibuat sebagai node paling depan (head) dari linked list.

Baris 5: Menandai bahwa fungsi berikutnya digunakan untuk mencetak isi linked list.
Baris 6: Mendefinisikan fungsi cetak_linked_list dengan parameter head.
Baris 7: Menginisialisasi variabel current untuk mulai menelusuri dari node pertama (head).
Baris 8: Mencetak label awal "Head →" untuk menunjukkan awal dari linked list.
Baris 9: Memulai perulangan selama current tidak bernilai None, artinya masih ada node yang bisa ditelusuri.
Baris 10: Mencetak nilai data dari node saat ini diikuti dengan tanda panah.
Baris 11: Berpindah ke node berikutnya dengan menggunakan nilai current['next'].
Baris 12: Setelah mencapai akhir linked list, mencetak "NULL" sebagai penutup.

Baris 13: Menandai bahwa bagian berikutnya digunakan untuk membuat linked list awal.
Baris 14: Menginisialisasi linked list dengan head = None, artinya list masih kosong.
Baris 15: Menyisipkan node dengan data 30 ke depan linked list kosong, sehingga menjadi head.
Baris 16: Menyisipkan node dengan data 20 ke depan linked list, menjadi node baru paling depan.
Baris 17: Menyisipkan node dengan data 10 ke depan linked list, sekarang menjadi head.

Baris 18: Menampilkan pesan bahwa isi linked list akan dicetak sebelum penyisipan node baru.
Baris 19: Memanggil fungsi cetak_linked_list untuk mencetak isi linked list saat ini.

Baris 20: Menandai bagian untuk penyisipan node baru.
Baris 21: Menyimpan nilai 99 ke dalam variabel data.
Baris 22: Menyisipkan node dengan data 99 ke depan linked list.

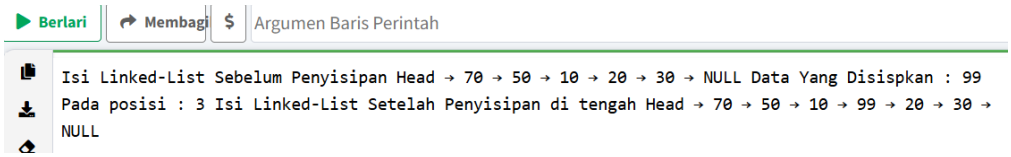
Baris 23: Mencetak nilai data yang baru saja disisipkan.

Baris 24: Menampilkan pesan bahwa isi linked list akan dicetak setelah penyisipan node baru.
Baris 25: Memanggil kembali fungsi cetak_linked_list untuk menampilkan isi terbaru dari linked list.

Praktik 25 :

```
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# sisip node diposisi mana saja
def sisip_dimana_aja(head, data, position):
    new_node = {'data': data, 'next': None}

    # cek jika posisi di awal pakai fungsi sisip_depan()
    if position == 0:
        return sisip_depan(head, data)

    current = head
    index = 0

    # ubah next dari node sebelumnya menjadi node baru
    new_node['next'] = current['next']
    current['next'] = new_node
    return head

## menampilkan linked-list
def cetak_linked_list(head):
    current = head
# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penyisipan")
cetak = cetak_linked_list(head)

# Penyisipan node
data = 99
pos = 3
head = sisip_dimana_aja(head, data, pos)

print("\nData Yang Disispkan : ", data)
print("Pada posisi : ", pos, "")

# cetak isi setelah penyisipan node baru di awal
print("\nIsi Linked-List Setelah Penyisipan di tengah")
cetak_linked_list(head)
```

Run :



Penjelasannya :

Baris 1: Menandai bahwa fungsi berikutnya digunakan untuk membuat node baru.
Baris 2: Mendefinisikan fungsi sisip_depan dengan dua parameter: head dan data.
Baris 3: Membuat node baru dalam bentuk dictionary dengan elemen 'data' dan 'next'. 'data' berisi nilai yang akan disimpan, sedangkan 'next' menunjuk ke node sebelumnya (head saat ini).
Baris 4: Mengembalikan node baru yang dibuat sebagai node paling depan (head) dari linked list.
Baris 6: Menandai bahwa fungsi berikutnya digunakan untuk menyisipkan node di posisi mana saja.
Baris 7: Mendefinisikan fungsi sisip_dimana_aja dengan parameter head, data, dan position.
Baris 8: Membuat node baru dengan data yang akan disisipkan dan nilai next diset ke None.
Baris 10: Mengecek apakah posisi penyisipan adalah 0.
Baris 11: Jika posisi 0, maka memanggil fungsi sisip_depan dan langsung mengembalikan hasilnya.
Baris 13: Menginisialisasi variabel current untuk mulai dari node pertama (head).
Baris 14: Menginisialisasi variabel index untuk melacak posisi saat ini dalam linked list.
Baris 16: Memulai perulangan untuk menuju node sebelum posisi penyisipan.
Baris 17: Memindahkan current ke node berikutnya.
Baris 18: Menambahkan nilai index sebanyak 1.
Baris 20: Jika current bernilai None, berarti posisi yang diberikan melebihi panjang linked list.
Baris 21: Menampilkan pesan peringatan bahwa posisi tidak valid.
Baris 22: Mengembalikan head tanpa perubahan.
Baris 24: Mengatur next dari node baru agar menunjuk ke node setelah current.
Baris 25: Mengatur next dari current agar menunjuk ke node baru.
Baris 26: Mengembalikan head yang telah diperbarui.
Baris 28: Menandai bahwa fungsi berikutnya digunakan untuk mencetak isi linked list.
Baris 29: Mendefinisikan fungsi cetak_linked_list dengan parameter head.
Baris 30: Menginisialisasi variabel current untuk mulai dari node pertama (head).
Baris 31: Menampilkan teks 'Head →' untuk menandai awal linked list.
Baris 32: Melakukan perulangan selama current tidak None.

Baris 33: Menampilkan nilai data dari node saat ini, diikuti panah.
Baris 34: Memindahkan current ke node berikutnya.
Baris 35: Menampilkan teks "NULL" sebagai penanda akhir dari linked list.
Baris 37: Menandai bagian penerapan program (implementasi linked list).
Baris 38: Inisialisasi linked list kosong dengan head = None.
Baris 39: Menyisipkan node 30 ke depan linked list.
Baris 40: Menyisipkan node 20 ke depan linked list.
Baris 41: Menyisipkan node 10 ke depan linked list.
Baris 42: Menyisipkan node 50 ke depan linked list.
Baris 43: Menyisipkan node 70 ke depan linked list.
Baris 45: Menampilkan keterangan bahwa linked list akan dicetak sebelum penyisipan node baru.
Baris 46: Memanggil fungsi cetak_linked_list untuk menampilkan isi linked list saat ini.
Baris 48: Menandai bahwa bagian ini adalah proses penyisipan node baru.
Baris 49: Menyimpan nilai 99 dalam variabel data.
Baris 50: Menyimpan nilai posisi 3 dalam variabel pos.
Baris 51: Memanggil fungsi sisip_dimana_aja untuk menyisipkan node 99 di posisi ke-3.
Baris 53: Menampilkan data yang disisipkan.
Baris 54: Menampilkan posisi tempat data disisipkan.
Baris 56: Menampilkan keterangan bahwa linked list akan dicetak setelah penyisipan node baru.
Baris 57: Memanggil fungsi cetak_linked_list untuk menampilkan isi linked list terbaru.

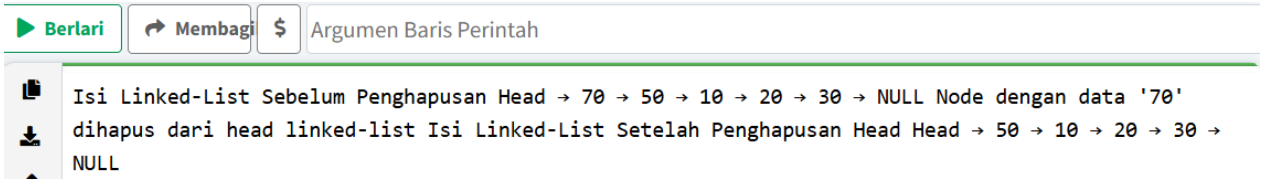
Praktik 26 :

```
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# sisip node diposisi mana saja
def sisip_dimana_aja(head, data, position):
    new_node = {'data': data, 'next': None}

    # cek jika posisi di awal pakai fungsi sisip_depan()
    if position == 0:
        return sisip_depan(head, data)

    current = head
    index = 0

    # traversal menuju posisi yang diinginkan dan bukan posisi 0
    while current is not None and index < position - 1:
        current = current['next']
        index += 1

    if current is None:
        print("Posisi melebihi panjang linked list!")
        return head
```

```
    # ubah next dari node sebelumnya menjadi node baru
    new_node['next'] = current['next']
    current['next'] = new_node
    return head

# menghapus head node dan mengembalikan head baru
def hapus_head(head):
    # cek apakah list kosong
    if head is None:
        print("Linked-List kosong, tidak ada yang bisa")
        return None
    print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
    return head['next']

## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")
```

```
# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30) # tail
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70) # head

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)

# Penghapusan head linked-list
head = hapus_head(head)

# cetak isi setelah hapus head linked-list
print("Isi Linked-List Setelah Penghapusan Head ")
cetak_linked_list(head)
```

Run :



Penjelasannya :

Baris 1: Menandai bahwa fungsi berikutnya digunakan untuk membuat node baru.
Baris 2: Mendefinisikan fungsi sisip_depan dengan dua parameter: head dan data.
Baris 3: Membuat node baru dalam bentuk dictionary dengan elemen 'data' dan 'next', di mana 'next' menunjuk ke node sebelumnya (head).
Baris 4: Mengembalikan node baru sebagai node paling depan (head) dari linked list.
Baris 6: Menandai bahwa fungsi berikutnya digunakan untuk menyisipkan node di posisi mana saja.
Baris 7: Mendefinisikan fungsi sisip_dimana_aja dengan parameter head, data, dan position.
Baris 8: Membuat node baru yang akan disisipkan, dengan data dan next di-set ke None.
Baris 10: Mengecek apakah posisi yang diberikan adalah 0.
Baris 11: Jika ya, memanggil fungsi sisip_depan dan langsung mengembalikan hasilnya.
Baris 13: Menginisialisasi variabel current untuk memulai traversal dari node pertama (head).
Baris 14: Menginisialisasi variabel index sebagai penghitung posisi saat traversal.
Baris 16: Melakukan perulangan untuk berpindah dari node satu ke node berikutnya hingga mencapai node sebelum posisi penyisipan.
Baris 17: Memindahkan current ke node berikutnya.
Baris 18: Menambahkan nilai index sebanyak 1.
Baris 20: Mengecek apakah current sudah None, artinya posisi melebihi panjang linked list.
Baris 21: Menampilkan pesan bahwa posisi melebihi panjang linked list.
Baris 22: Mengembalikan head tanpa melakukan penyisipan.
Baris 24: Menyambungkan node baru ke node setelah current.
Baris 25: Menyambungkan node sebelumnya (current) ke node baru.
Baris 26: Mengembalikan head yang telah diperbarui.
Baris 28: Menandai bahwa fungsi berikut digunakan untuk menghapus node head dan mengembalikan head baru.
Baris 29: Mendefinisikan fungsi hapus_head dengan parameter head.
Baris 30: Mengecek apakah linked list kosong.
Baris 31: Jika kosong, menampilkan pesan bahwa tidak ada yang bisa dihapus.
Baris 32: Mengembalikan None karena linked list kosong.
Baris 33: Menampilkan data node head yang akan dihapus.

Baris 34: Mengembalikan node berikutnya sebagai head baru dari linked list.
Baris 36: Menandai bahwa fungsi berikut digunakan untuk mencetak isi linked list.
Baris 37: Mendefinisikan fungsi cetak_linked_list dengan parameter head.
Baris 38: Menginisialisasi pointer current dengan node pertama (head).
Baris 39: Mencetak teks 'Head →' sebagai awalan tampilan linked list.
Baris 40: Melakukan perulangan selama current tidak kosong (None).
Baris 41: Menampilkan data dari node saat ini, diikuti tanda panah.
Baris 42: Memindahkan current ke node berikutnya.
Baris 43: Menampilkan teks 'NULL' sebagai penanda akhir dari linked list.
Baris 45: Menandai bagian penerapan program.
Baris 46: Inisialisasi linked list kosong (head = None).
Baris 47: Menyisipkan node 30 ke depan linked list (ini akan menjadi tail).
Baris 48: Menyisipkan node 20 ke depan linked list.
Baris 49: Menyisipkan node 10 ke depan linked list.
Baris 50: Menyisipkan node 50 ke depan linked list.
Baris 51: Menyisipkan node 70 ke depan linked list (ini akan menjadi head).

Baris 53: Menampilkan keterangan bahwa linked list akan dicetak sebelum proses penghapusan.
Baris 54: Memanggil fungsi cetak_linked_list untuk menampilkan isi awal linked list.
Baris 56: Melakukan penghapusan node head dengan memanggil hapus_head.
Baris 57: Menampilkan isi linked list setelah node head dihapus.
Baris 58: Memanggil fungsi cetak_linked_list untuk mencetak isi linked list yang baru.

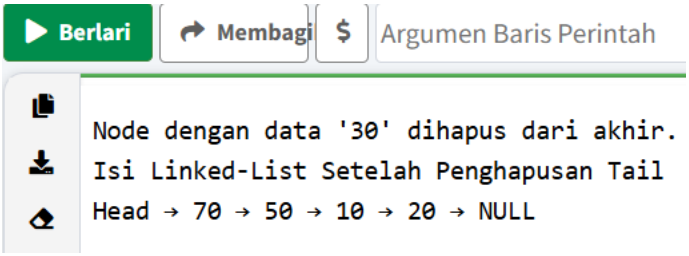
Praktik 27 :

```
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menghapus head node dan mengembalikan head baru
def hapus_tail(head):
    # cek apakah head node == None
    if head is None:
        print('Linked-List Kosong, tidak ada yang bisa dihapus!')
        return None

    # cek node hanya 1
    if head['next'] is None:
        print(f"Node dengan data '{head['data']}' dihapus. Linked list sekarang kosong.")
        return None

    current = head
    while current['next']['next'] is not None:
        current = current['next']

    print(f"\nNode dengan data '{current['next']['data']}' dihapus dari akhir.")
    current['next'] = None
    return head
```

```
## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30) # tail
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70) # head

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)
```

```
# Penghapusan tail linked-list
head = hapus_tail(head)

# cetak isi setelah hapus Tail linked-list
print("Isi Linked-List Setelah Penghapusan Tail ")
cetak_linked_list(head)
```

Run :



Penjelasannya :

Baris 1: Menandai bahwa fungsi berikutnya digunakan untuk membuat node baru.
Baris 2: Mendefinisikan fungsi sisip_depan dengan dua parameter: head dan data.
Baris 3: Membuat node baru dalam bentuk dictionary dengan elemen 'data' dan 'next', di mana 'next' menunjuk ke node sebelumnya (head).
Baris 4: Mengembalikan node baru sebagai node paling depan (head) dari linked list.
Baris 6: Menandai bahwa fungsi berikutnya digunakan untuk menghapus node terakhir (tail).
Baris 7: Mendefinisikan fungsi hapus_tail dengan parameter head.
Baris 8: Mengecek apakah linked list kosong (head bernilai None).
Baris 9: Jika kosong, mencetak pesan bahwa tidak ada yang bisa dihapus.
Baris 10: Mengembalikan None karena linked list kosong.
Baris 12: Mengecek apakah hanya ada satu node (node pertama tidak memiliki next).
Baris 13: Jika ya, mencetak pesan bahwa node tersebut dihapus dan linked list menjadi kosong.
Baris 14: Mengembalikan None karena sekarang tidak ada lagi node.
Baris 16: Menginisialisasi pointer current untuk mulai dari node pertama (head).
Baris 17: Melakukan perulangan hingga current['next']['next'] adalah None, artinya current ada di node sebelum tail.
Baris 18: Memindahkan current ke node berikutnya.
Baris 20: Mencetak pesan bahwa node tail dengan data tertentu akan dihapus.
Baris 21: Menjadikan next dari node sebelum tail sebagai None, memutus node tail.
Baris 22: Mengembalikan head dari linked list setelah penghapusan.
Baris 24: Menandai bahwa fungsi berikut digunakan untuk mencetak isi linked list.
Baris 25: Mendefinisikan fungsi cetak_linked_list dengan parameter head.
Baris 26: Menginisialisasi pointer current dengan node pertama (head).
Baris 27: Menampilkan teks 'Head →' sebagai awalan tampilan linked list.
Baris 28: Melakukan perulangan selama current tidak kosong (None).
Baris 29: Menampilkan data dari node saat ini, diikuti tanda panah.
Baris 30: Memindahkan current ke node berikutnya.
Baris 31: Menampilkan teks 'NULL' sebagai penanda akhir dari linked list.

Baris 33: Menandai bagian penerapan program.
Baris 34: Inisialisasi linked list kosong (head = None).
Baris 35: Menyisipkan node 30 ke depan linked list (ini akan menjadi tail).
Baris 36: Menyisipkan node 20 ke depan linked list.
Baris 37: Menyisipkan node 10 ke depan linked list.
Baris 38: Menyisipkan node 50 ke depan linked list.
Baris 39: Menyisipkan node 70 ke depan linked list (ini akan menjadi head).
Baris 41: Menampilkan teks bahwa linked list akan dicetak sebelum proses penghapusan.
Baris 42: Memanggil fungsi cetak_linked_list untuk menampilkan isi awal linked list.
Baris 44: Memanggil fungsi hapus_tail untuk menghapus node terakhir dari linked list.
Baris 45: Menampilkan teks bahwa linked list akan dicetak setelah proses penghapusan tail.
Baris 46: Memanggil fungsi cetak_linked_list untuk menampilkan isi linked list setelah penghapusan tail.

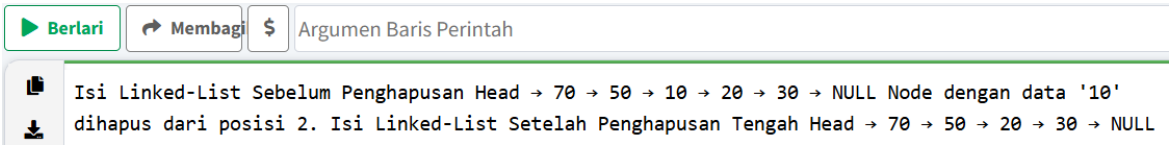
Praktik 28 :

```
# Praktek 28 : Menghapus node di posisi manapun (tengah)
# membuat node baru
def sisip_depan(head, data):
    new_node = {'data': data, 'next': head}
    return new_node

# menghapus head node dan mengembalikan head baru
def hapus_head(head):
    # cek apakah list kosong
    if head is None:
        print("Linked-List kosong, tidak ada yang bisa")
        return None
    print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
    return head['next']

# menghapus node pada posisi manapun (tengah)
def hapus_tengah(head, position):
    # cek apakah head node == None
    if head is None:
        print('\nLinked-List Kosong, tidak ada yang bisa dihapus!')
        return None
```

```
    # cek apakah posisi < 0
    if position < 0:
        print('\nPosisi Tidak Valid')
        return head

    # Cek apakah posisi = 0
    if position == 0:
        print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.")
        hapus_head(head)
        return head['next']

    current = head
    index = 0

    # cari node sebelum posisi target
    while current is not None and index < position -1:
        current = current['next']
        index += 1

    # Jika posisi yang diinputkan lebih besar dari panjang list
    if current is None or current['next'] is None:
        print("\nPosisi melebih panjang dari linked-list")
        return head

    print(f"\nNode dengan data '{current['next']['data']}' dihapus dari posisi {position}.")
    current['next'] = current['next']['next']
    return head

## menampilkan linked-list
def cetak_linked_list(head):
    current = head
    print('Head', end=' → ')
    while current is not None:
        print(current['data'], end=' → ')
        current = current['next']
    print("NULL")

# Penerapan
# membuat linked-list awal
head = None
head = sisip_depan(head, 30) # tail
head = sisip_depan(head, 20)
head = sisip_depan(head, 10)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70) # head
```

```
# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)

# Penghapusan ditengah linked-list
head = hapus_tengah(head, 2)

# cetak isi setelah hapus tengah linked-list
print("\nIsi Linked-List Setelah Penghapusan Tengah ")
cetak_linked_list(head)
```

Run :



Penjelasannya :

Baris 1: Komentar penanda judul praktik.
Baris 2: Komentar bahwa fungsi berikutnya digunakan untuk membuat node baru.
Baris 3: Mendefinisikan fungsi sisip_depan dengan parameter head dan data.
Baris 4: Membuat node baru dalam bentuk dictionary dengan 'data' dan 'next' yang menunjuk ke head.
Baris 5: Mengembalikan node baru sebagai head yang baru dari linked list.
Baris 7: Komentar bahwa fungsi berikutnya digunakan untuk menghapus node dari posisi paling depan (head).
Baris 8: Mendefinisikan fungsi hapus_head dengan parameter head.
Baris 9: Mengecek apakah head bernilai None (linked list kosong).
Baris 10: Jika kosong, mencetak pesan bahwa tidak ada yang bisa dihapus.
Baris 11: Mengembalikan None.
Baris 12: Mencetak pesan node head yang dihapus.
Baris 13: Mengembalikan node berikutnya sebagai head baru.

Baris 15: Komentar bahwa fungsi berikutnya digunakan untuk menghapus node di posisi tertentu.
Baris 16: Mendefinisikan fungsi hapus_tengah dengan parameter head dan position.
Baris 17: Mengecek apakah head kosong.
Baris 18: Jika kosong, mencetak pesan dan mengembalikan None.
Baris 20: Mengecek jika position kurang dari 0.
Baris 21: Jika tidak valid, mencetak pesan dan mengembalikan head tanpa perubahan.
Baris 23: Mengecek jika posisi adalah 0 (artinya head yang dihapus).
Baris 24: Mencetak pesan bahwa node di posisi 0 akan dihapus.
Baris 25: Memanggil fungsi hapus_head, tapi hasilnya tidak digunakan karena tidak disimpan ke variabel.
Baris 26: Mengembalikan head['next'] sebagai head baru.
Baris 28: Inisialisasi pointer current ke head dan variabel indeks index ke 0.
Baris 29: Perulangan untuk mencari node sebelum posisi target.
Baris 30: Memindahkan current ke node berikutnya.
Baris 31: Menambahkan indeks saat berpindah node.
Baris 33: Mengecek apakah posisi yang diminta melebihi panjang linked list.

Baris 34: Jika ya, mencetak pesan dan mengembalikan head tanpa perubahan.
Baris 36: Mencetak data node yang akan dihapus di posisi tertentu.
Baris 37: Mengatur next dari node sebelumnya untuk langsung menunjuk ke node setelah target, sehingga node target dihapus.
Baris 38: Mengembalikan head.
Baris 40: Komentar bahwa fungsi berikut digunakan untuk mencetak isi linked list.
Baris 41: Mendefinisikan fungsi cetak_linked_list dengan parameter head.
Baris 42: Inisialisasi pointer current ke node pertama.
Baris 43: Mencetak teks 'Head → ' untuk menandai awal tampilan linked list.
Baris 44: Melakukan perulangan selama current tidak None.
Baris 45: Menampilkan data node saat ini.
Baris 46: Memindahkan pointer ke node berikutnya.
Baris 47: Menampilkan 'NULL' sebagai penanda akhir linked list.
Baris 49: Komentar bagian penerapan program.
Baris 50: Inisialisasi linked list kosong (head = None).
Baris 51: Menyisipkan node 30 (akan menjadi tail).
Baris 52: Menyisipkan node 20 ke depan.
Baris 53: Menyisipkan node 10 ke depan.
Baris 54: Menyisipkan node 50 ke depan.
Baris 55: Menyisipkan node 70 ke depan (menjadi head).

Baris 57: Menampilkan pesan sebelum penghapusan node.
Baris 58: Memanggil fungsi untuk mencetak isi linked list.

Baris 60: Memanggil fungsi hapus_tengah untuk menghapus node pada posisi ke-2.
Baris 62: Menampilkan pesan setelah penghapusan node tengah.
Baris 63: Mencetak kembali isi linked list setelah node tengah dihapus.