

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 :

```
Linked-List :  
Head → 10 → 11 → 12 → NULL  
  
** Process exited - Return Code: 0 **  
Press Enter to exit terminal
```

PENJELASAN :

Baris 1

Komentar: memberi tahu bahwa fungsi di bawah akan membuat node.

Baris 2–3

Fungsi untuk membuat node baru. Node ini berisi data dan penunjuk ke node berikutnya (masih kosong atau None).

Baris 4

Komentar: menjelaskan bahwa fungsi berikut akan menambahkan node di akhir list.

Baris 5–6

Fungsi membuat node baru dari data yang diberikan.

Baris 7–8

Jika linked list masih kosong (belum ada head), node baru langsung jadi head.

Baris 9–11

Jika list tidak kosong, lakukan perulangan untuk mencari node terakhir.

Baris 12

Setelah ketemu node terakhir, hubungkan node baru di bagian akhirnya.

Baris 13

Kembalikan head supaya linked list tetap bisa diakses dari awal.

Baris 14

Komentar: memberitahu bahwa fungsi berikut akan mencetak isi linked list.

Baris 15–16

Mulai dari head, siapkan variabel untuk berjalan ke node-node berikutnya.

Baris 17–19

Selama masih ada node, cetak data node, lalu lanjut ke node berikutnya.

Baris 20

Setelah semua node dicetak, tampilkan "NULL" untuk menunjukkan akhir list.

Baris 21

Inisialisasi head sebagai kosong, berarti list belum punya node.

Baris 22–24

Menambahkan tiga node dengan nilai 10, 11, dan 12 ke dalam list.

Baris 25

Tampilkan teks "Linked-List:" ke layar.

Baris 26

Cetak semua isi linked list dari awal sampai akhir.

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

# Penerapan
head = None
head = tambah_node(head, 10)
head = tambah_node(head, 15)
head = tambah_node(head, 117)
head = tambah_node(head, 19)
```






```
# 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 :

	Isi Linked-List
	Head → 10 → 15 → 117 → 19 → NULL
	Jumlah Nodes = 4
	HEAD Node : 10
	TAIL Node : 19

PENJELASAN :

Baris 1

Komentar: Memberi tahu bahwa fungsi berikut digunakan untuk membuat sebuah node.

Baris 2–3

Fungsi untuk membuat node baru. Node ini terdiri dari dua bagian: satu menyimpan data, satu lagi menunjuk ke node berikutnya (masih kosong).

Baris 4

Komentar: Menjelaskan bahwa fungsi berikut akan menambahkan node di akhir list.

Baris 5–6

Node baru dibuat menggunakan data yang diberikan.

Baris 7–8

Jika list masih kosong, node baru langsung dijadikan sebagai head.

Baris 9–11

Jika tidak kosong, cari node terakhir dalam list dengan cara berjalan satu per satu.

Baris 12

Setelah ketemu node terakhir, sambungkan node baru ke bagian akhir list.

Baris 13

Kembalikan head agar list tetap bisa diakses dari awal.

Baris 14

Komentar: Memberi tahu bahwa fungsi berikut dipakai untuk mencetak isi linked list.

Baris 15–16

Mulai dari head, siapkan pointer untuk menelusuri node-node.

Baris 17–19

Selama node masih ada, cetak isi datanya dan lanjut ke node berikutnya.

Baris 20

Setelah semua node dicetak, tampilkan tulisan "NULL" sebagai penanda akhir list.

Baris 21

Komentar: Fungsi berikut digunakan untuk menghitung jumlah elemen dalam list.

Baris 22–23

Inisialisasi penghitung, lalu mulai dari head.

Baris 24–26

Selama masih ada node, tambahkan penghitung dan lanjut ke node berikutnya.

Baris 27

Kembalikan jumlah total node yang ditemukan.

Baris 28

Komentar: Fungsi berikut digunakan untuk mencari node terakhir (tail).

Baris 29–30

Jika list kosong, kembalikan kosong juga.

Baris 31–33

Jika list tidak kosong, telusuri hingga node terakhir (yang next-nya kosong).

Baris 34

Kembalikan node terakhir (tail).

Baris 36

Inisialisasi list sebagai kosong (belum punya node/head).

Baris 37–40

Menambahkan empat node ke dalam list, dengan nilai 10, 15, 117, dan 19.

Baris 42

Cetak teks “Isi Linked-List” ke layar.

Baris 43

Panggil fungsi untuk mencetak isi semua node.

Baris 45

Cetak teks “Jumlah Nodes =” ke layar.

Baris 46

Panggil fungsi untuk menghitung jumlah node dan tampilkan hasilnya.

Baris 48

Cetak teks "HEAD Node :." ke layar.

Baris 49

Tampilkan nilai dari node pertama (head).

Baris 51

Cetak teks "TAIL Node :." ke layar.

Baris 52

Panggil fungsi untuk mendapatkan node terakhir dan tampilkan datanya.

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
```

```

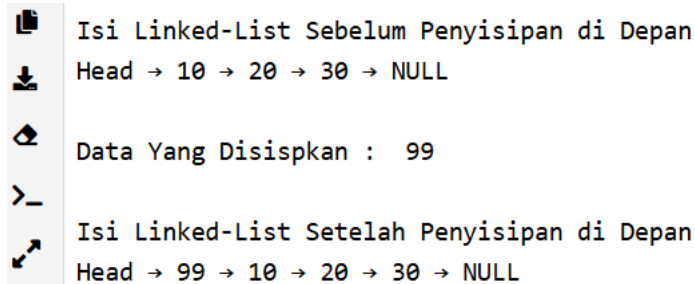
data = 99
head = sisip_depan(head, data)

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

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

```

HASIL :



```

Isi Linked-List Sebelum Penyisipan di Depan
Head → 10 → 20 → 30 → NULL

Data Yang Disisipkan : 99

Isi Linked-List Setelah Penyisipan di Depan
Head → 99 → 10 → 20 → 30 → NULL

```

PENJELASAN :

Baris 1

Komentar: Memberi tahu bahwa fungsi berikut akan digunakan untuk membuat dan menyisipkan node baru di bagian depan (awal) linked list.

Baris 2–3

Fungsi membuat node baru dengan data yang diberikan. Node baru ini langsung menunjuk ke node pertama yang sudah ada (head lama), sehingga otomatis jadi node pertama yang baru. Fungsi lalu mengembalikan node baru ini sebagai head baru dari list.

Baris 5

Komentar: Menjelaskan bahwa fungsi berikut akan digunakan untuk menampilkan isi linked list.

Baris 6–10

Mulai dari head, fungsi menelusuri node satu per satu sambil mencetak isi datanya. Setelah semua data dicetak, ditampilkan tulisan "NULL" untuk menandakan akhir list.

Baris 12

Inisialisasi linked list dengan head kosong.

Baris 13–15

Menyisipkan tiga node baru satu per satu di depan. Karena selalu disisipkan di depan, urutannya akan menjadi:
10 (paling akhir), lalu 20, lalu 30 (paling depan).

Baris 17

Menampilkan teks penjelasan bahwa isi linked list akan ditampilkan sebelum ada penyisipan baru.

Baris 18

Memanggil fungsi cetak untuk menampilkan isi linked list saat ini.

Baris 20

Menyimpan nilai 99 ke dalam variabel sebagai data yang akan disisipkan.

Baris 21

Menyisipkan node dengan data 99 ke depan linked list. Node ini akan menjadi node pertama yang baru.

Baris 23

Menampilkan teks yang menunjukkan data yang disisipkan ke depan list.

Baris 25

Menampilkan teks bahwa isi linked list akan dicetak setelah penyisipan.

Baris 26

Mencetak ulang isi linked list, kali ini dengan node baru (99) di paling depan.

PRAKTEK 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

    # 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

## 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)
head = sisip_depan(head, 50)
head = sisip_depan(head, 70)

# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penvisipan")

```

```

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)

```

HASIL :

	Isi Linked-List Sebelum Penyisipan
	Head → 70 → 50 → 10 → 20 → 30 → NULL
	Data Yang Disisipkan : 99
	Pada posisi : 3
	Isi Linked-List Setelah Penyisipan di tengah
	Head → 70 → 50 → 10 → 99 → 20 → 30 → NULL

PENJELASAN :

Baris 1

Komentar: Menjelaskan bahwa fungsi berikut dipakai untuk menyisipkan node di bagian depan (head).

Baris 2–3

Fungsi membuat node baru yang langsung menunjuk ke node awal yang lama, lalu node baru itu dikembalikan sebagai head baru.

Baris 5

Komentar: Fungsi ini menyisipkan node di posisi tertentu, bukan hanya di depan.

Baris 6

Membuat node baru dengan data yang diberikan, dan penunjuk berikutnya masih kosong.

Baris 8–9

Jika posisi yang diminta adalah nol (artinya di awal), maka cukup panggil fungsi untuk menyisipkan di depan.

Baris 11–12

Menyiapkan variabel untuk mulai traversal (penelusuran) dari node pertama dan posisi awal indeks.

Baris 14–16

Melakukan perulangan untuk menelusuri node satu per satu sampai mencapai node sebelum posisi yang diminta.

Baris 18–20

Jika sudah mencapai akhir list sebelum mencapai posisi yang diminta, berarti posisi tidak valid — muncul peringatan, dan linked list tidak diubah.

Baris 22–23

Jika posisi valid, node baru dimasukkan ke dalam list:

- Node baru menunjuk ke node setelahnya.
- Node sebelumnya menunjuk ke node baru.

Baris 24

Kembalikan head agar tetap bisa diakses dari awal list.

Baris 26

Komentar: Fungsi untuk menampilkan isi linked list.

Baris 27–31

Fungsi mencetak isi linked list dari node pertama sampai node terakhir, ditutup dengan “NULL”.

Baris 33

Inisialisasi linked list masih kosong.

Baris 34–38

Menyisipkan lima node di awal secara bertahap. Karena semua disisipkan di depan, urutan akhirnya dari atas ke bawah adalah:

70 → 50 → 10 → 20 → 30

Baris 40–41

Menampilkan isi linked list sebelum penyisipan di posisi tertentu.

Baris 43–44

Menyisipkan nilai 99 ke posisi ke-3. (Ingat, posisi dimulai dari nol. Jadi disisipkan setelah node ketiga.)

Baris 46–47

Menampilkan nilai data yang disisipkan dan posisi penyisipannya.

Baris 49–50

Menampilkan isi linked list setelah node baru disisipkan di tengah.

PRAKTEK 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)

```


HASIL :




Isi Linked-List Sebelum Penghapusan




Head → 70 → 50 → 10 → 20 → 30 → NULL



Node dengan data '70' dihapus dari head linked-list



Isi Linked-List Setelah Penghapusan Head



Head → 50 → 10 → 20 → 30 → NULL

PENJELASAN :

Baris 1

Komentar: Fungsi berikut digunakan untuk menyisipkan node baru di bagian depan list.

Baris 2–3

Node baru dibuat dan langsung diarahkan ke node lama yang sebelumnya jadi head. Lalu node baru ini dijadikan head yang baru.

Baris 5

Komentar: Fungsi ini digunakan untuk menyisipkan node di posisi mana pun dalam list.

Baris 6

Membuat node baru, dengan data yang diberikan, dan penunjuk ke node selanjutnya masih kosong.

Baris 8–9

Jika posisi yang diminta adalah posisi nol (paling depan), cukup gunakan fungsi penyisipan di depan.

Baris 11–12

Siapkan pointer untuk mulai traversal dari node pertama, dan variabel untuk melacak posisi.

Baris 14–16

Lakukan penelusuran hingga mencapai node sebelum posisi yang diinginkan.

Baris 18–20

Jika tidak menemukan posisi yang valid (misalnya posisi lebih panjang dari isi list), tampilkan peringatan dan tidak ubah isi list.

Baris 22–23

Jika posisi valid:

- Node baru diarahkan ke node setelahnya.
- Node sebelumnya diarahkan ke node baru itu.

Baris 24

Kembalikan head supaya list tetap bisa diakses dari awal.

Baris 26

Komentar: Fungsi berikut digunakan untuk menghapus node head dari linked list.

Baris 27–29

Jika linked list kosong, tampilkan pesan bahwa tidak ada yang bisa dihapus, dan kembalikan kosong.

Baris 30–31

Jika tidak kosong, tampilkan data yang dihapus, dan kembalikan node setelah head sebagai head yang baru.

Baris 33

Komentar: Fungsi ini untuk mencetak isi linked list.

Baris 34–38

Mulai dari head, cetak satu per satu isi datanya, hingga akhir (NULL).

Baris 40

Inisialisasi linked list kosong.

Baris 41–45

Tambahkan lima node ke depan list secara berurutan.

Karena selalu disisipkan di depan, urutan akhirnya akan seperti ini:

70 → 50 → 10 → 20 → 30

(70 jadi head, 30 jadi tail)

Baris 47–48

Tampilkan isi linked list sebelum penghapusan head.

Baris 50

Hapus node pertama (head) dan kembalikan node berikutnya sebagai head baru.

Baris 52–53

Tampilkan isi linked list setelah penghapusan head. Sekarang, node pertama (70) sudah tidak ada.

PRAKTEK 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)
```

HASIL :



Isi Linked-List Sebelum Penghapusan



Head → 70 → 50 → 10 → 20 → 30 → NULL



Node dengan data '30' dihapus dari akhir.



Isi Linked-List Setelah Penghapusan Tail



Head → 70 → 50 → 10 → 20 → NULL

PENJELASAN :

Baris 1

Komentar: Fungsi ini akan menyisipkan node di bagian depan (head).

Baris 2–3

Membuat node baru dan langsung menghubungkannya ke node yang sebelumnya jadi head, lalu mengembalikannya sebagai head baru.

Baris 5

Komentar: Fungsi ini akan menghapus tail atau node terakhir dari linked list.

Baris 6–8

Jika linked list kosong, tampilkan pesan dan kembalikan None.

Baris 10–12

Jika isi linked list hanya satu node, maka node tersebut dihapus dan linked list jadi kosong. Tampilkan pesan dan kembalikan None.

Baris 14

Siapkan pointer untuk mulai dari node pertama (head).

Baris 15–16

Telusuri node satu per satu sampai menemukan node sebelum node terakhir (kedua dari belakang).

Baris 18

Tampilkan data node terakhir yang akan dihapus.

Baris 19

Putuskan koneksi ke node terakhir dengan mengatur next dari node sebelumnya menjadi None.

Baris 20

Kembalikan head agar linked list tetap bisa diakses dari awal.

Baris 22

Komentar: Fungsi ini akan menampilkan isi linked list.

Baris 23–27

Telusuri dan cetak setiap node dari head sampai akhir, diakhiri dengan tulisan "NULL".

Baris 29

Inisialisasi linked list masih kosong.

Baris 30–34

Tambahkan node satu per satu di depan. Karena penambahan dilakukan di depan, urutannya akan menjadi:

70 → 50 → 10 → 20 → 30

(70 jadi head, 30 jadi tail)

Baris 36–37

Tampilkan isi linked list sebelum penghapusan tail.

Baris 39

Lakukan penghapusan node terakhir (tail).

Baris 41–42

Tampilkan isi linked list setelah penghapusan tail. Sekarang node 30 sudah tidak ada.

PRAKTEK 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 melebihi 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

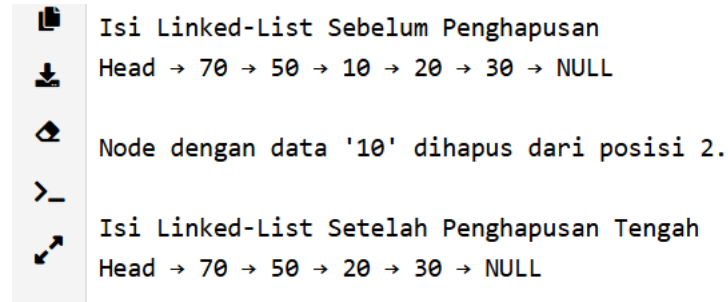
# cetak isi linked-list awal
print("Isi Linked-List Sebelum Penghapusan")
cetak_linked_list(head)

# 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)
```

HASIL :



```
Isi Linked-List Sebelum Penghapusan
Head → 70 → 50 → 10 → 20 → 30 → NULL

Node dengan data '10' dihapus dari posisi 2.

Isi Linked-List Setelah Penghapusan Tengah
Head → 70 → 50 → 20 → 30 → NULL
```

PENJELASAN :

Baris 1

Komentar: Memberi tahu bahwa ini adalah contoh menghapus node di posisi mana pun (biasanya di tengah).

Baris 2–3

Fungsi untuk menambahkan node baru di bagian depan (head), langsung diarahkan ke node sebelumnya.

Baris 5–10

Fungsi untuk menghapus node pertama (head).

Kalau list kosong, tampilkan pesan.

Kalau tidak, hapus node head dan kembalikan node setelahnya sebagai head baru.

Baris 12

Komentar: Fungsi utama untuk menghapus node di posisi tertentu (termasuk tengah).

Baris 13–15

Kalau list kosong, tampilkan pesan dan keluar.

Baris 17–19

Kalau posisi yang diminta kurang dari nol, tampilkan pesan posisi tidak valid.

Baris 21–24

Kalau posisi = 0, berarti hapus head. Tampilkan pesan, panggil fungsi penghapusan head, dan kembalikan node berikutnya.

Baris 26–27

Mulai traversal dari head, dan gunakan indeks untuk menghitung posisi.

Baris 29–31

Telusuri list sampai ke node sebelum node yang ingin dihapus.

Baris 33–35

Jika posisi lebih besar dari jumlah node (tidak ketemu node target), tampilkan pesan peringatan dan tidak menghapus.

Baris 37–38

Tampilkan data node yang akan dihapus, lalu ubah next dari node sebelumnya untuk melewati node target (jadi seperti dihapus).

Baris 39

Kembalikan head agar list tetap bisa digunakan dari awal.

Baris 41–45

Fungsi untuk mencetak isi linked list: dari head sampai akhir (NULL).

Baris 47–52

Buat linked list dengan lima node, disisipkan dari depan.

Urutannya akan:

70 → 50 → 10 → 20 → 30

Baris 54–55

Cetak isi list sebelum node dihapus.

Baris 57

Lakukan penghapusan node di posisi ke-2 (indeks dimulai dari 0, jadi node ke-3 secara visual).

Dalam list saat ini, node di posisi 2 adalah 10.

Baris 59–60

Cetak isi list setelah node 10 dihapus.

Hasil akhirnya menjadi:

70 → 50 → 20 → 30