

NAMA : Septia Agus Rianto  
NIM : 24241022  
MATKUL : STRUKTUR DATA

---

### Praktek 22

```
1 # function untuk membuat node
2 def buat_node(data):
3     return {'data': data, 'next': None}
4
5 # menambahkan node di akhir list
6 def tambah_node(head, data):
7     new_node = buat_node(data)
8     if head is None:
9         return new_node
10    current = head
11    while current['next'] is not None:
12        current = current['next']
13    current['next'] = new_node
14    return head
15
16 # menampilkan linked-list
17 def cetak_linked_list(head):
18     current = head
19     print('Head', end=' → ')
20     while current is not None:
21         print(current['data'], end=' → ')
22         current = current['next']
23     print("NULL")
24
25 # Contoh Penerapan
26 # Head awal dari linked-list
27 head = None
28
29 # Tambah node
30 head = tambah_node(head, 10)
31 head = tambah_node(head, 11)
32 head = tambah_node(head, 12)
33
34 # cetak linked-list
35 print('Linked-List : ')
36 cetak_linked_list(head)
```

### Outputnya :



Linked-List :



Head → 10 → 11 → 12 → NULL

### Penjelasannya :

1. `def buat_node(data):`  
Membuat fungsi untuk membuat node baru.
2. `return {'data': data, 'next': None}`  
Mengembalikan node berupa dictionary dengan data dan penunjuk `next`.

3. `def tambah_node(head, data):`  
Fungsi untuk menambahkan node di akhir linked list.
4. `new_node = buat_node(data)`  
Membuat node baru dari data yang diberikan.
5. `if head is None:`  
Cek apakah linked list masih kosong.
6. `return new_node`  
Jika kosong, node baru menjadi head.
7. `current = head`  
Mulai penelusuran dari head.
8. `while current['next'] is not None:`  
Telusuri sampai node terakhir.
9. `current = current['next']`  
Pindah ke node berikutnya.
10. `current['next'] = new_node`  
Sambungkan node baru ke akhir list.
11. `return head`  
Kembalikan head sebagai awal list.
12. `def cetak_linked_list(head):`  
Fungsi untuk mencetak isi linked list.
13. `current = head`  
Mulai dari head.
14. `print('Head', end=' → ')`  
Cetak kata "Head" di awal.
15. `while current is not None:`  
Selama masih ada node, lanjutkan.
16. `print(current['data'], end=' → ')`  
Cetak data dari node.
17. `current = current['next']`  
Pindah ke node berikutnya.
18. `print("NULL")`  
Akhiri cetakan dengan "NULL".
19. `head = None`  
Awal linked list masih kosong.
20. `head = tambah_node(head, 10)`  
Tambah node dengan data 10.
21. `head = tambah_node(head, 11)`  
Tambah node dengan data 11.
22. `head = tambah_node(head, 12)`  
Tambah node dengan data 12.
23. `print('Linked-List : ')`  
Cetak judul linked list.
24. `cetak_linked_list(head)`  
Tampilkan isi linked list.

### Praktek 23

```
1 # function untuk membuat node
2 def buat_node(data):
3     return {'data': data, 'next': None}
4
5 # menambahkan node di akhir list
6 def tambah_node(head, data):
7     new_node = buat_node(data)
8     if head is None:
9         return new_node
10    current = head
11    while current['next'] is not None:
12        current = current['next']
13    current['next'] = new_node
14    return head
15
16 # traversal untuk cetak isi linked-list
17 def traversal_to_display(head):
18     current = head
19     print('Head', end=' → ')
20     while current is not None:
21         print(current['data'], end=' → ')
22         current = current['next']
23     print("NULL")
24
25 # traversal untuk menghitung jumlah elemen dalam linked-list
26 def traversal_to_count_nodes(head):
27     count = 0
28     current = head
29     while current is not None:
30         count += 1
31         current = current['next']
32     return count
33
34 # traversal untuk mencari dimana tail (node terakhir)
35 def traversal_to_get_tail(head):
36     if head is None:
37         return None
38     current = head
39     while current['next'] is not None:
40         current = current['next']
41     return current
42
43 # Penerapan
44 head = None
45 head = tambah_node(head, 10)
46 head = tambah_node(head, 15)
47 head = tambah_node(head, 117)
48 head = tambah_node(head, 19)
49
50 # cetak isi linked-list
51 print("Isi Linked-List")
52 traversal_to_display(head)
53
54 # cetak jumlah node
55 print("Jumlah Nodes = ", traversal_to_count_nodes(head))
56
57 # cetak HEAD node
58 print("HEAD Node : ", head['data'])
59
60 # cetak TAIL NODE
61 print("TAIL Node : ", traversal_to_get_tail(head)['data'])
```

**Outputnya :**

```

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

```

### Penjelasannya :

1. `def buat_node(data):`  
Buat fungsi untuk membuat node baru.
2. `return {'data': data, 'next': None}`  
Kembalikan node dengan data dan penunjuk ke `None`.
3. `def tambah_node(head, data):`  
Buat fungsi untuk menambahkan node di akhir linked list.
4. `new_node = buat_node(data)`  
Buat node baru dari data.
5. `if head is None:`  
Jika linked list kosong...
6. `return new_node`  
...kembalikan node baru sebagai head.
7. `current = head`  
Mulai penelusuran dari head.
8. `while current['next'] is not None:`  
Ulangi hingga sampai node terakhir.
9. `current = current['next']`  
Pindah ke node berikutnya.
10. `current['next'] = new_node`  
Tambahkan node baru di akhir.
11. `return head`  
Kembalikan head.
12. `def traversal_to_display(head):`  
Fungsi untuk mencetak isi linked list.
13. `current = head`  
Mulai dari head.
14. `print('Head', end=' → ')`  
Cetak "Head →" sebagai awalan.
15. `while current is not None:`  
Selama masih ada node...
16. `print(current['data'], end=' → ')`  
Cetak data dari node.
17. `current = current['next']`  
Lanjut ke node berikutnya.
18. `print("NULL")`  
Tampilkan akhir linked list.
19. `def traversal_to_count_nodes(head):`  
Fungsi untuk menghitung jumlah node.

```

20. count = 0
    Inisialisasi counter.
21. current = head
    Mulai dari head.
22. while current is not None:
    Selama node masih ada...
23. count += 1
    Tambah hitungan.
24. current = current['next']
    Lanjut ke node berikutnya.
25. return count
    Kembalikan jumlah node.

26. def traversal_to_get_tail(head):
    Fungsi untuk mencari node terakhir.
27. if head is None:
    Jika linked list kosong...
28. return None
    ...tidak ada tail.
29. current = head
    Mulai dari head.
30. while current['next'] is not None:
    Ulangi hingga sampai node terakhir.
31. current = current['next']
    Lanjut ke node berikutnya.
32. return current
    Kembalikan node terakhir (tail).

33. head = None
    Awalnya linked list kosong.
34. head = tambah_node(head, 10)
    Tambah node dengan data 10.
35. head = tambah_node(head, 15)
    Tambah node dengan data 15.
36. head = tambah_node(head, 117)
    Tambah node dengan data 117.
37. head = tambah_node(head, 19)
    Tambah node dengan data 19.

38. print("Isi Linked-List")
    Cetak judul tampilan.
39. traversal_to_display(head)
    Tampilkan isi linked list.

40. print("Jumlah Nodes = ", traversal_to_count_nodes(head))
    Cetak jumlah node.

41. print("HEAD Node : ", head['data'])
    Cetak data dari node pertama.

```

```
42. print("TAIL Node : ", traversal_to_get_tail(head) ['data'])
    Cetak data dari node terakhir.
```

## Praktek 24

```
1  # membuat node baru
2  def sisip_depan(head, data):
3      new_node = {'data': data, 'next': head}
4      return new_node
5
6  # menampilkan linked-list
7  def cetak_linked_list(head):
8      current = head
9      print('Head', end=' → ')
10     while current is not None:
11         print(current['data'], end=' → ')
12         current = current['next']
13     print("NULL")

15 # Penerapan membuat linked-list awal
16 head = None
17 head = sisip_depan(head, 30)
18 head = sisip_depan(head, 20)
19 head = sisip_depan(head, 10)
20
21 # cetak isi linked-list awal
22 print("Isi Linked-List Sebelum Penyisipan di Depan")
23 cetak = cetak_linked_list(head)
24
25 # Penyisipan node
26 data = 99
27 head = sisip_depan(head, data)

25 # Penyisipan node
26 data = 99
27 head = sisip_depan(head, data)
28
29 print("\nData Yang Disisipkan : ", data)
30
31 # cetak isi setelah penyisipan node baru di awal
32 print("\nIsi Linked-List Setelah Penyisipan di Depan")
33 cetak_linked_list(head)
```

Outputnya :

```
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
```

### Penjelasannya:

1. `def sisip_depan(head, data):`  
Definisikan fungsi untuk menyisipkan node di depan.
2. `new_node = {'data': data, 'next': head}`  
Buat node baru yang menunjuk ke head lama.
3. `return new_node`  
Kembalikan node baru sebagai head baru.
4. `def cetak_linked_list(head):`  
Definisikan fungsi untuk mencetak linked list.
5. `current = head`  
Mulai dari node pertama (head).
6. `print('Head', end=' → ')`  
Cetak teks awalan "Head →".
7. `while current is not None:`  
Selama masih ada node...
8. `print(current['data'], end=' → ')`  
Cetak data dari node saat ini.
9. `current = current['next']`  
Pindah ke node berikutnya.
10. `print("NULL")`  
Cetak akhir linked list.
11. `head = None`  
Inisialisasi linked list kosong.
12. `head = sisip_depan(head, 30)`  
Tambah node 30 di depan.
13. `head = sisip_depan(head, 20)`  
Tambah node 20 di depan.
14. `head = sisip_depan(head, 10)`  
Tambah node 10 di depan.
15. `print("Isi Linked-List Sebelum Penyisipan di Depan")`  
Tampilkan judul untuk linked list awal.
16. `cetak = cetak_linked_list(head)`  
Cetak linked list sebelum penyisipan baru.
17. `data = 99`  
Siapkan data yang akan disisipkan.
18. `head = sisip_depan(head, data)`  
Tambahkan data 99 di depan.
19. `print("\nData Yang Disisipkan : ", data)`  
Cetak data yang disisipkan.
20. `print("\nIsi Linked-List Setelah Penyisipan di Depan")`  
Judul tampilan setelah penyisipan.
21. `cetak_linked_list(head)`  
Cetak isi linked list setelah node 99 disisipkan.

## Praktek 25

```
1 # membuat node baru
2 def sisip_depan(head, data):
3     new_node = {'data': data, 'next': head}
4     return new_node
5 # sisip node diposisi mana saja
6 def sisip_dimana_aja(head, data, position):
7     new_node = {'data': data, 'next': None}
8     # cek jika posisi di awal pakai fungsi sisip_depan()
9     if position == 0:
10         return sisip_depan(head, data)
11     current = head
12     index = 0
13     # traversal menuju posisi yang diinginkan dan bukan posisi 0
14     while current is not None and index < position - 1:
15         current = current['next']
16         index += 1
17     if current is None:
18         print("Posisi melebihi panjang linked list!")
19         return head
20     # ubah next dari node sebelumnya menjadi node baru
21     new_node['next'] = current['next']
22     current['next'] = new_node
23     return head
24 ## menampilkan linked-list
25 def cetak_linked_list(head):
26     current = head
27     print('Head', end=' → ')
28     while current is not None:
29         print(current['data'], end=' → ')
30         current = current['next']
31     print("NULL")
32 # Penerapan
33 # membuat linked-list awal
34 head = None
35 head = sisip_depan(head, 30)
36 head = sisip_depan(head, 20)
37 head = sisip_depan(head, 10)
38 head = sisip_depan(head, 50)
39 head = sisip_depan(head, 70)
40 # cetak isi linked-list awal
41 print("Isi Linked-List Sebelum Penyisipan")
42 cetak = cetak_linked_list(head)
43 # Penyisipan node
44 data = 99
45 pos = 3
46 head = sisip_dimana_aja(head, data, pos)
47 print("\nData Yang Disisipkan : ", data)
48 print("Pada posisi : ", pos, "")
49 # cetak isi setelah penyisipan node baru di awal
50 print("\nIsi Linked-List Setelah Penyisipan di tengah")
51 cetak_linked_list(head)
```



## Outputnya :

```
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
```

## Penjelasannya :

1. `def sisip_depan(head, data):`  
Buat fungsi untuk menambah node di depan.
2. `new_node = {'data': data, 'next': head}`  
Node baru menunjuk ke head lama.
3. `return new_node`  
Kembalikan node baru sebagai head baru.
4. `def sisip_dimana_aja(head, data, position):`  
Fungsi untuk menyisipkan node di posisi tertentu.
5. `new_node = {'data': data, 'next': None}`  
Buat node baru.
6. `if position == 0:`  
Jika posisi 0 (awal)...
7. `return sisip_depan(head, data)`  
Sisipkan di depan.
8. `current = head`  
Mulai dari head.
9. `index = 0`  
Inisialisasi posisi.
10. `while current is not None and index < position - 1:`  
Traversal hingga posisi sebelum target.
11. `current = current['next']`  
Pindah ke node berikutnya.
12. `index += 1`  
Tambah posisi.
13. `if current is None:`  
Jika posisi tidak valid...
14. `print("Posisi melebihi panjang linked list!")`  
Cetak pesan kesalahan.
15. `return head`  
Tidak disisipkan, kembalikan head.
16. `new_node['next'] = current['next']`  
Sambungkan node baru ke node setelahnya.
17. `current['next'] = new_node`  
Sambungkan node sebelumnya ke node baru.
18. `return head`  
Kembalikan head.
19. `def cetak_linked_list(head):`  
Fungsi untuk mencetak linked list.

```

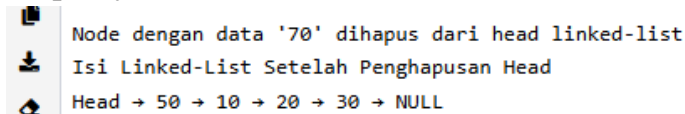
20. current = head
    Mulai dari head.
21. print('Head', end=' → ')
    Cetak awalan.
22. while current is not None:
    Selama ada node...
23. print(current['data'], end=' → ')
    Cetak data node.
24. current = current['next']
    Pindah ke node berikutnya.
25. print("NULL")
    Akhir linked list.
26. head = None
    Inisialisasi list kosong.
27. head = sisip_depan(head, 30)
    Tambah 30 di depan.
28. head = sisip_depan(head, 20)
    Tambah 20 di depan.
29. head = sisip_depan(head, 10)
    Tambah 10 di depan.
30. head = sisip_depan(head, 50)
    Tambah 50 di depan.
31. head = sisip_depan(head, 70)
    Tambah 70 di depan.
32. print("Isi Linked-List Sebelum Penyisipan")
    Tampilkan judul tampilan awal.
33. cetak = cetak_linked_list(head)
    Cetak isi linked list.
34. data = 99
    Data yang ingin disisipkan.
35. pos = 3
    Posisi penyisipan.
36. head = sisip_dimana_aja(head, data, pos)
    Sisipkan 99 di posisi ke-3.
37. print("\nData Yang Disisipkan : ", data)
    Cetak data yang disisipkan.
38. print("Pada posisi : ", pos, "")
    Cetak posisi penyisipan.
39. print("\nIsi Linked-List Setelah Penyisipan di tengah")
    Judul tampilan akhir.
40. cetak_linked_list(head)
    Cetak isi linked list setelah disisipkan.

```

## Praktek 26

```
1 # membuat node baru
2 def sisip_depan(head, data):
3     new_node = {'data': data, 'next': head}
4     return new_node
5 # sisip node diposisi mana saja
6 def sisip_dimana_aja(head, data, position):
7     new_node = {'data': data, 'next': None}
8     # cek jika posisi di awal pakai fungsi sisip_depan()
9     if position == 0:
10         return sisip_depan(head, data)
11     current = head
12     index = 0
13     # traversal menuju posisi yang diinginkan dan bukan posisi 0
14     while current is not None and index < position - 1:
15         current = current['next']
16         index += 1
17     if current is None:
18         print("Posisi melebihi panjang linked list!")
19         return head
20     # ubah next dari node sebelumnya menjadi node baru
21     new_node['next'] = current['next']
22     current['next'] = new_node
23     return head
24 # menghapus head node dan mengembalikan head baru
25 def hapus_head(head):
26     # cek apakah list kosong
27     if head is None:
28         print("Linked-List kosong, tidak ada yang bisa")
29         return None
30     print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
31     return head['next']
32 ## menampilkan linked-list
33 def cetak_linked_list(head):
34     current = head
35     print('Head', end=' → ')
36     while current is not None:
37         print(current['data'], end=' → ')
38         current = current['next']
39     print("NULL")
40 # Penerapan
41 # membuat linked-list awal
42 head = None
43 head = sisip_depan(head, 30) # tail
44 head = sisip_depan(head, 20)
45 head = sisip_depan(head, 10)
46 head = sisip_depan(head, 50)
47 head = sisip_depan(head, 70) # head
48 # cetak isi linked-list awal
49 print("Isi Linked-List Sebelum Penghapusan")
50 cetak_linked_list(head)
51 # Penghapusan head linked-list
52 head = hapus_head(head)
53 # cetak isi setelah hapus head linked-list
54 print("Isi Linked-List Setelah Penghapusan Head ")
55 cetak_linked_list(head)
```

## Outputnya :



```
Node dengan data '70' dihapus dari head linked-list
Isi Linked-List Setelah Penghapusan Head
Head → 50 → 10 → 20 → 30 → NULL
```

## Penjelasannya :

1. `def sisip_depan(head, data):`  
Fungsi untuk menambah node di depan.
2. `new_node = {'data': data, 'next': head}`  
Node baru menunjuk ke head lama.
3. `return new_node`  
Kembalikan node baru sebagai head baru.
4. `def sisip_dimana_aja(head, data, position):`  
Fungsi untuk menyisipkan node di posisi tertentu.
5. `new_node = {'data': data, 'next': None}`  
Buat node baru.
6. `if position == 0:`  
Jika posisi 0...
7. `return sisip_depan(head, data)`  
Sisipkan di depan.
8. `current = head`  
Mulai traversal dari head.
9. `index = 0`  
Inisialisasi indeks.
10. `while current is not None and index < position - 1:`  
Telusuri hingga sebelum posisi tujuan.
11. `current = current['next']`  
Pindah ke node berikutnya.
12. `index += 1`  
Tambah indeks.
13. `if current is None:`  
Jika posisi melebihi panjang list...
14. `print("Posisi melebihi panjang linked list!")`  
Cetak pesan error.
15. `return head`  
Kembalikan head tanpa perubahan.
16. `new_node['next'] = current['next']`  
Hubungkan node baru ke node setelahnya.
17. `current['next'] = new_node`  
Hubungkan node sebelumnya ke node baru.
18. `return head`  
Kembalikan head baru.
19. `def hapus_head(head):`  
Fungsi untuk menghapus node pertama (head).
20. `if head is None:`  
Cek jika list kosong.

```

21. print("Linked-List kosong, tidak ada yang bisa")
    Cetak pesan error.
22. return None
    Kembalikan None karena tidak ada yang dihapus.
23. print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-
    list")
    Tampilkan data yang dihapus.
24. return head['next']
    Kembalikan node berikutnya sebagai head baru.

25. def cetak_linked_list(head):
    Fungsi untuk mencetak isi linked list.
26. current = head
    Mulai dari node pertama.
27. print('Head', end=' → ')
    Cetak awalan.
28. while current is not None:
    Selama masih ada node...
29. print(current['data'], end=' → ')
    Cetak data node.
30. current = current['next']
    Pindah ke node selanjutnya.
31. print("NULL")
    Akhiri cetakan.

32. head = None
    Inisialisasi linked list kosong.

33-37. head = sisip_depan(head, ...)
Tambah node 30, 20, 10, 50, 70 (hasil akhir: 70 → 50 → 10 → 20 → 30).

38. print("Isi Linked-List Sebelum Penghapusan")
    Judul cetakan awal.
39. cetak_linked_list(head)
    Cetak isi linked list sebelum penghapusan.
40. head = hapus_head(head)
    Hapus node paling depan (70).
41. print("Isi Linked-List Setelah Penghapusan Head ")
    Judul tampilan sesudah penghapusan.
42. cetak_linked_list(head)
    Cetak isi list setelah head dihapus (hasil: 50 → 10 → 20 → 30).

```

## Praktek 27

```
1 # membuat node baru
2 def sisip_depan(head, data):
3     new_node = {'data': data, 'next': head}
4     return new_node
5 # menghapus head node dan mengembalikan head baru
6 def hapus_tail(head):
7     # cek apakah head node == None
8     if head is None:
9         print('Linked-List Kosong, tidak ada yang bisa dihapus!')
10        return None
11    # cek node hanya 1
12    if head['next'] is None:
13        print(f"Node dengan data '{head['data']}' dihapus. Linked list sekarang kosong.")
14        return None
15    current = head
16    while current['next']['next'] is not None:
17        current = current['next']
18    print(f"\nNode dengan data '{current['next']['data']}' dihapus dari akhir.")
19    current['next'] = None
20    return head
21 ## menampilkan linked-list
22 def cetak_linked_list(head):
23     current = head
24     print('Head', end=' → ')
25     while current is not None:
26         print(current['data'], end=' → ')
27         current = current['next']
28     print("NULL")
29 # Penerapan
30 # membuat linked-list awal
31 head = None
32 head = sisip_depan(head, 30) # tail
33 head = sisip_depan(head, 20)
34 head = sisip_depan(head, 10)
35 head = sisip_depan(head, 50)
36 head = sisip_depan(head, 70) # head
37 # cetak isi linked-list awal
38 print("Isi Linked-List Sebelum Penghapusan")
39 cetak_linked_list(head)
40 # Penghapusan tail linked-list
41 head = hapus_tail(head)
42 # cetak isi setelah hapus Tail linked-list
43 print("Isi Linked-List Setelah Penghapusan Tail ")
44 cetak_linked_list(head)
```

### Outputnya :

```
Node dengan data '30' dihapus dari akhir.
Isi Linked-List Setelah Penghapusan Tail
Head → 70 → 50 → 10 → 20 → NULL
```

### Penjelasannya :

1. def sisip\_depan(head, data):  
Fungsi menambahkan node di depan linked list.

2. `new_node = {'data': data, 'next': head}`  
Node baru menunjuk ke head lama.
3. `return new_node`  
Kembalikan node baru sebagai head baru.
  
4. `def hapus_tail(head):`  
Fungsi untuk menghapus node terakhir (tail).
5. `if head is None:`  
Jika list kosong...
6. `print('Linked-List Kosong, tidak ada yang bisa dihapus!')`  
Cetak pesan error.
7. `return None`  
Tidak ada node untuk dihapus.
8. `if head['next'] is None:`  
Jika hanya 1 node...
9. `print(f"Node dengan data '{head['data']}' dihapus. Linked list sekarang kosong.")`  
Tampilkan data node yang dihapus.
10. `return None`  
Linked list jadi kosong.
11. `current = head`  
Mulai traversal dari head.
12. `while current['next']['next'] is not None:`  
Telusuri hingga sebelum node terakhir.
13. `current = current['next']`  
Pindah ke node berikutnya.
14. `print(f"\nNode dengan data '{current['next']['data']}' dihapus dari akhir.")`  
Cetak data node terakhir yang akan dihapus.
15. `current['next'] = None`  
Putuskan hubungan dengan node terakhir (hapus).
16. `return head`  
Kembalikan head baru.
  
17. `def cetak_linked_list(head):`  
Fungsi untuk menampilkan isi linked list.
18. `current = head`  
Mulai dari head.
19. `print('Head', end=' → ')`  
Cetak awalan.
20. `while current is not None:`  
Selama masih ada node...
21. `print(current['data'], end=' → ')`  
Cetak data node.
22. `current = current['next']`  
Pindah ke node berikutnya.
23. `print("NULL")`  
Akhir tampilan linked list.

24. head = None  
Inisialisasi linked list kosong.

25–29. head = sisip\_depan(head, ...)  
Tambahkan node satu per satu: 30, 20, 10, 50, 70.  
Hasil akhirnya: 70 → 50 → 10 → 20 → 30

30. print("Isi Linked-List Sebelum Penghapusan")  
Judul sebelum penghapusan.

31. cetak\_linked\_list(head)  
Cetak isi linked list.

32. head = hapus\_tail(head)  
Hapus node terakhir (30).

33. print("Isi Linked-List Setelah Penghapusan Tail ")  
Judul setelah penghapusan.

34. cetak\_linked\_list(head)  
Cetak hasil setelah tail dihapus.

## Praktek 28

```
1 # Praktek 28 : Menghapus node di posisi manapun (tengah)
2 # membuat node baru
3 def sisip_depan(head, data):
4     new_node = {'data': data, 'next': head}
5     return new_node
6 # menghapus head node dan mengembalikan head baru
7 def hapus_head(head):
8     # cek apakah list kosong
9     if head is None:
10         print("Linked-List kosong, tidak ada yang bisa")
11         return None
12     print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")
13     return head['next']
14 # menghapus node pada posisi manapun (tengah)
15 def hapus_tengah(head, position):
16     # cek apakah head node == None
17     if head is None:
18         print('\nLinked-List Kosong, tidak ada yang bisa dihapus!')
19         return None
20     # cek apakah posisi < 0
21     if position < 0:
22         print('\nPosisi Tidak Valid')
23         return head
24     # Cek apakah posisi = 0
25     if position == 0:
26         print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.")
```



```

27     hapus_head(head)
28     return head['next']
29 current = head
30 index = 0
31 # cari node sebelum posisi target
32 while current is not None and index < position -1:
33     current = current['next']
34     index += 1
35 # Jika posisi yang diinputkan lebih besar dari panjang list
36 if current is None or current['next'] is None:
37     print("\nPosisi melebihi panjang dari linked-list")
38     return head
39 print(f"\nNode dengan data '{current['next']['data']}' dihapus dari posisi {position}.")

40     current['next'] = current['next']['next']
41     return head
42 ## menampilkan linked-list
43 def cetak_linked_list(head):
44     current = head
45     print('Head', end=' → ')
46     while current is not None:
47         print(current['data'], end=' → ')
48         current = current['next']
49     print("NULL")
50 # Penerapan
51 # membuat linked-list awal
52 head = None

53 head = sisip_depan(head, 30) # tail
54 head = sisip_depan(head, 20)
55 head = sisip_depan(head, 10)
56 head = sisip_depan(head, 50)
57 head = sisip_depan(head, 70) # head
58 # cetak isi linked-list awal
59 print("Isi Linked-List Sebelum Penghapusan")
60 cetak_linked_list(head)
61 # Penghapusan ditengah linked-list
62 head = hapus_tengah(head, 2)
63 # cetak isi setelah hapus tengah linked-list
64 print("\nIsi Linked-List Setelah Penghapusan Tengah ")
65 cetak_linked_list(head)

```

### Outputnya :

```

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

```

### Penjelasannya :

1. `def sisip_depan(head, data):`  
Fungsi menambah node di depan linked list.
2. `new_node = {'data': data, 'next': head}`  
Buat node baru yang menunjuk ke head lama.

3. `return new_node`  
Kembalikan node baru sebagai head baru.
4. `def hapus_head(head):`  
Fungsi hapus node pertama (head).
5. `if head is None:`  
Cek apakah linked list kosong.
6. `print("Linked-List kosong, tidak ada yang bisa")`  
Jika kosong, tampilkan pesan.
7. `return None`  
Kembalikan None karena list kosong.
8. `print(f"\nNode dengan data '{head['data']}' dihapus dari head linked-list")`  
Tampilkan data node yang dihapus.
9. `return head['next']`  
Kembalikan node berikutnya sebagai head baru.
10. `def hapus_tengah(head, position):`  
Fungsi hapus node di posisi mana saja selain head.
11. `if head is None:`  
Cek list kosong.
12. `print('\nLinked-List Kosong, tidak ada yang bisa dihapus!')`  
Pesan jika kosong.
13. `return None`  
Kembalikan None.
14. `if position < 0:`  
Cek posisi valid (tidak negatif).
15. `print('\nPosisi Tidak Valid')`  
Tampilkan pesan posisi tidak valid.
16. `return head`  
Kembalikan list tanpa perubahan.
17. `if position == 0:`  
Jika posisi 0, hapus head.
18. `print(f"Node dengan data '{head['data']}' dihapus dari posisi 0.")`  
Tampilkan data node yang dihapus.
19. `hapus_head(head)`  
Panggil fungsi hapus head.
20. `return head['next']`  
Kembalikan head baru setelah hapus.
21. `current = head`  
Mulai traversal dari head.
22. `index = 0`  
Inisialisasi indeks.
23. `while current is not None and index < position - 1:`  
Cari node sebelum posisi target.
24. `current = current['next']`  
Pindah ke node berikutnya.
25. `index += 1`  
Tambah indeks.

```

26. if current is None or current['next'] is None:
    Jika posisi melebihi panjang list.
27. print("\nPosisi melebihi panjang dari linked-list")
    Tampilkan pesan error.
28. return head
    Kembalikan list tanpa perubahan.
29. print(f"\nNode dengan data '{current['next']['data']}' dihapus dari
    posisi {position}.")
    Tampilkan data node yang akan dihapus.
30. current['next'] = current['next']['next']
    Lewati node target, hapus dari linked list.
31. return head
    Kembalikan head list.

32. def cetak_linked_list(head):
    Fungsi cetak isi linked list.
33. current = head
    Mulai dari head.
34. print('Head', end=' → ')
    Cetak awalan.
35. while current is not None:
    Selama node ada.
36. print(current['data'], end=' → ')
    Cetak data node.
37. current = current['next']
    Pindah ke node berikutnya.
38. print("NULL")
    Akhiri cetakan.

39. head = None
    Inisialisasi linked list kosong.

```

40–44. head = sisip\_depan(head, ...)  
 Tambah node 30, 20, 10, 50, 70 → jadi 70 → 50 → 10 → 20 → 30

```

45. print("Isi Linked-List Sebelum Penghapusan")
    Judul cetakan awal.
46. cetak_linked_list(head)
    Cetak isi linked list sebelum hapus.
47. head = hapus_tengah(head, 2)
    Hapus node di posisi 2 (nilai 10).
48. print("\nIsi Linked-List Setelah Penghapusan Tengah ")
    Judul cetakan sesudah hapus.
49. cetak_linked_list(head)
    Cetak linked list setelah hapus (hasil: 70 → 50 → 20 → 30).

```

