STRUKTUR DATA
TEORI 6

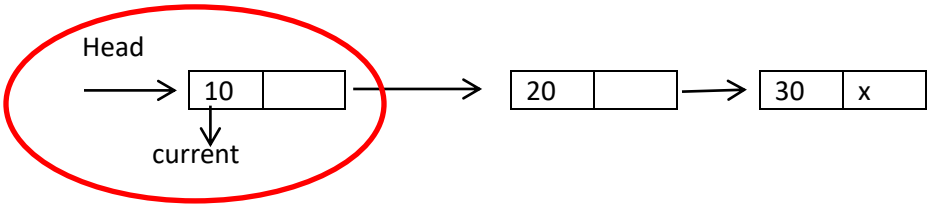
Nama	: Dhiaka Shabrina Assyifa
NIM	: A11.2020.13094
Kelompok	: A11.4312

Buat rangkuman mengenai Single Linked List 2 pada materi ini: https://youtu.be/pqj05_A3DLo
Dan sebutkan:

1. *Apa yang membedakan menambahkan data di depan dan tengah ?*
2. *Apa yang lebih optimal menghapus data dibelakang atau didepan, beri penjelasan!*

Kumpulkan dalam bentuk pdf, boleh diketik atau tulis tangan.
Penjelasan

1. *Apa yang membedakan menambahkan data di depan dan tengah ?*
   Penjelasan :
   Yang membedakan untuk penambahan data di depan dan di tengah menurut saya adalah pada penambahan data di tengah membuat pointer baru yang menunjuk ke node yang ditunjuk oleh head yaitu current untuk mencari node sebelum node baru.



2. *Apa yang lebih optimal menghapus data di belakang atau di depan, beri penjelasan!*
   Penjelasan :
   Jika ditanya mana yang lebih optimal, maka saya menjawab menghapus data di depan, karena lebih cepat dan mudah. Apabila menghapus data di belakang maka harus menambahkan pointer baru yaitu current. Current untuk menempati data sebelum tail untuk selanjutnya Next_Node dari current diisi next_node dari tail kemudia tail menunjuk node current.
   Nb: lebih jelas lihat uraian di bawah ini pada bagian hapus data

3. *Penjelasan Youtube*
   Single Linked List terdiri dari head dan tail dan sisanya adalah rangkaian dari data data tersebut. Satu data terdiri dari node, node berisi data itu sendiri dan pointernya.

   A. Tipe data dinamis-SLL

   - Array vs Linked list

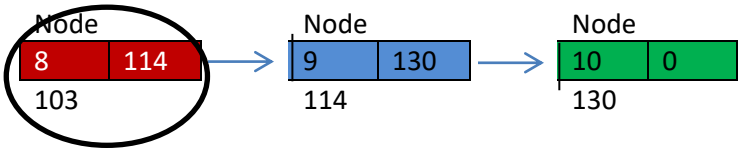
Array	Linked List
Statis	Dinamis
Penambahan / penghapusan data terbatas	Penambahan / pengurangan data tidak terbatas
Random access	Sequential access
Penghapusan array tidak mungkin	Penghapusan linked list mudah

   Linked list adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling sambung-menyambung, dinamis dan terbatas. Tiap elemen dihubungkan ke elemen yang lain dengan bantuan variabel pointer. Pointer adalah alamat elemen. Linked list sering disebut sebagai senarai berantai. Masing-masing data

dalam Linked list disebut dengan node (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa struct yang terdiri dari beberapa field.

- Alokasi memori
- SLL non circular
  - Pembentukan SLLNC

```
class Node(object):
    def_init_(self,data=None,next_node=None):
self.data=data
self.next_node=next_node
```



Pembuatan kelas bernama Node yang beisi 2 field, yaitu field data bertipe integer dan field next_node yang bertipe pointer dari node

Nb: Node 8 terhubung dengan node 9 karena 114 berasal dari node dengan data 9

B. Penambahan Data

➢ Penambahan data di depan

Hal yang harus diperhatikan

1. Apakah Linked list masih kosong

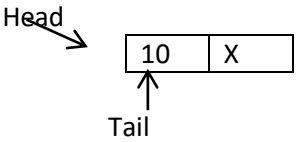
   Linked masih kosong

   Maka pointer:

   Head = None

   Tail = None

   a. Buat node baru
   b. Masukkan datanya(misal:10)
   c. Jika linked list masih kosong(ditandai dengan head is none), maka head dan tail menunjuk node baru tersebut

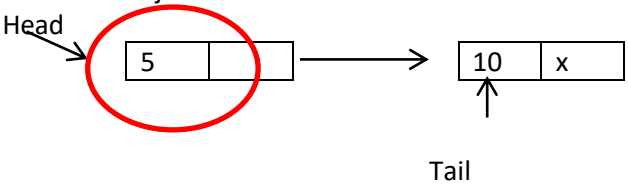


2. Apakah Linked list sudah ada datanya

   Linked sudah ada isinya

   Memastikan linked list sudah ada datanya atau tidak

   a. Buat node baru
   b. Masukkan datanya(misal:5)
   c. Jika linked list tidak kosong(ditandai dengan head is not none), maka:
      a) Next dari node baru menunjuk ke head sebelumnya
      b) Head menunjuk ke node baru



- Penambahan data di belakang

  Hal yang harus diperhatikan

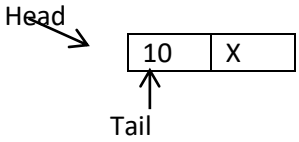
  1. Apakah Linked list masih kosong? Linked masih kosong

     Maka pointer:

     Head = None

     Tail = None

     a. Buat node baru
     b. Masukkan datanya(misal:10)
     c. Jika linked list masih kosong(ditandai dengan head is none), maka head dan tail menunjuk node baru tersebut

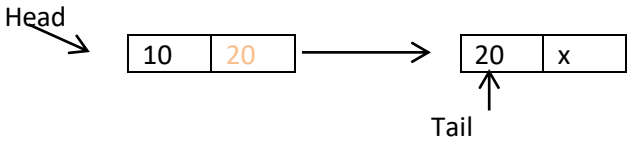


  2. Apakah Linked list sudah ada datanya

     Linked sudah ada isinya

     Memastikan linked list sudah ada datanya atau tidak

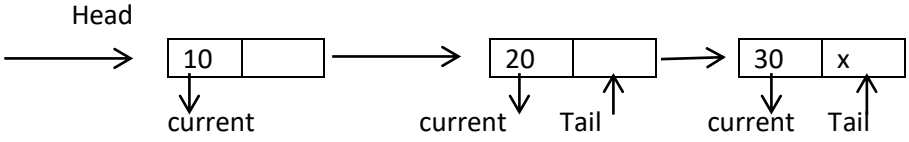
     a. Buat node baru
     b. Masukkan datanya(misal:20)
     c. Jika linked list tidak kosong(ditandai dengan Head is not None), maka:
        a) Next dari node baru menunjuk ke head sebelumnya
        b) Tail menunjuk ke node baru



- Penambahan data di tengah
  - implementasi fungsi pada python

```
#menambah data di tengah pada list
def tambahtengah(self,data):
    #membuat pointer baru menunjuk ke node yang ditunjuk oleh HEAD
    current = self.head
    found = False
    #perulangan mencari node yang dicari
    while current and found is False:
        if current.get_data()==data:
            found=True
            obj1=str(input("Masukkan data yang ingin anda
Tambahkan: "))
            new_node = Node(obj1)
            new_node.next_node = current.next_node
            current.next_node= new_node
        else:
            current=current.get_next()
    return found
```

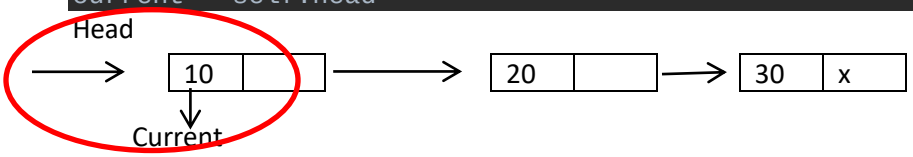
  - modifikasi dari mencari data



- misalkan menambah data setelah 20
  a. deklarasikan penunjuk node sekarang(current)menunjuk ke head

```
current = self.head
```

Head



  b. selama node current tidak kosong

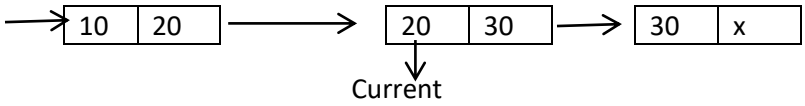
```
#perulangan mencari node yang dicari
while current and found is False:
```

  1) cek apakah 20 sama dengan data dari node current?

```
if current.get_data()==data:
    found=True
    obj1=str(input("Masukkan data yang ingin anda Tambahkan:
"))
    new_node = Node(obj1)
    new_node.next_node = current.next_node
    current.next_node= new_node
else:
    current=current.get_next()
```

Tidak maka masuk ke kondisi else, sehingga current pindah ke next

Head



Cek lagi apakah 20 sama dengan data dari node current?

```
if current.get_data()==data:
    found=True
    obj1=str(input("Masukkan data yang ingin anda Tambahkan:
"))
    new_node = Node(obj1)
    new_node.next_node = current.next_node
    current.next_node= new_node
```

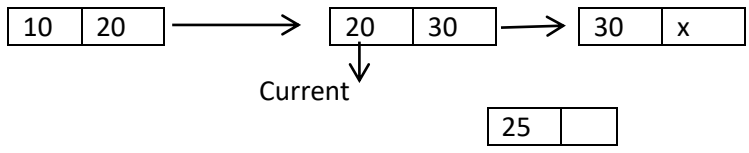
Ya, benar. Maka :

  a) buat node baru

```
obj1=str(input("Masukkan data yang ingin anda Tambahkan:
"))
new_node = Node(obj1)
```

  b) isikan data baru (misalkan 25)

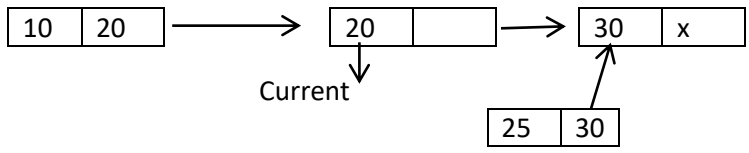
Head



  c) next dari node baru menunjuk next dari current

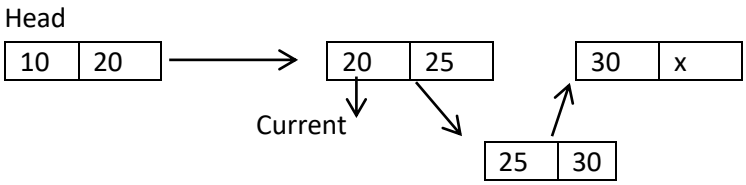
```
new_node.next_node = current.next_node
```

Head



d) next dari current menunjuk node baru

```
current.next_node= new_node
```



2) pointer current menunjuk node setelahnya

C. Menampilkan Data

- Menampilkan data di belakang

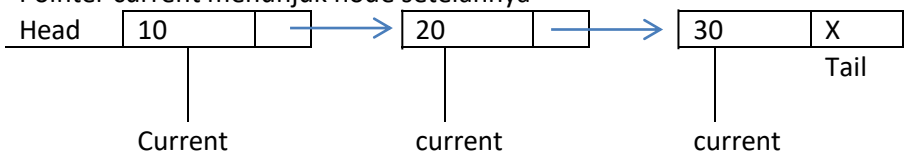
Ada dua hal yang harus diketahui:

1. Apakah linked list masih kosong

   Jika linked list masih kosng, maka tidak akan menampilkan data apapun

   ```
   if (selfhead is None):
           print("Data masih kosong")
   ```

2. Apakah linked list sudah ada datanya
   a. Deklarasikan penunjuk node sekarang(current) menunjuk ke head
   b. Selama node current tidak kosong
      1) Tampilkan datanya
      2) Pointer current menunjuk node setelahnya



D. Menghapus Data

- Menghapus data di depan

Hal yang harus diperhatikan

1. Apakah Linked list masih kosong

   Linked list masih kosong
   a. Cek apakah list kosong
   b. Jika ya maka tampilan “Data Tidak Ada”

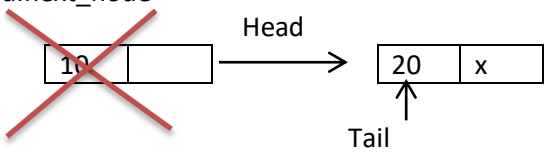
   ```
   if (selfhead is None):
           print("Data Tidak Ada")
   ```

2. Apakah Linked list sudah ada datanya

   Linked list sudah ada isinya

   Pindahkan Head ke Node setelah Head

   Selfhead = self.head.next_node



- Menghapus data di belakang

  Hal yang harus diperhatikan

  1. Apakah Linked list masih kosong

     Linked list masih kosong

     a. Cek apakah list kosong
     b. Jika ya maka tampilan “Data Tidak Ada”

        if (selfhead is None):

            print("Data Tidak Ada")

  2. Apakah Linked list sudah ada datanya

     Linked list sudah ada isinya

     implementasi fungsi pada python

```
#menghapus data di belakang dari list
def deletedibelakang(self):
    current_node = self.head
    if (self.head is None):
        print("Data masih kosong")
    else:
        while current_node.next_node.next_node is not None:
            current_node= current_node.next_node

            current_node.next_node=self.tail.next_node
            self.tail=current_node
```

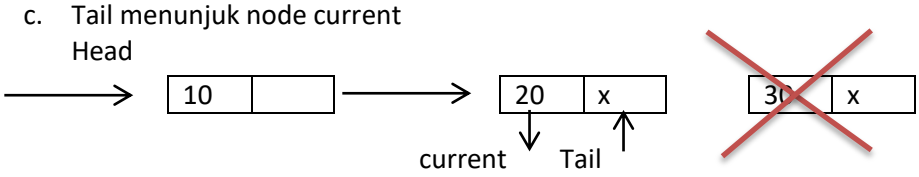
     a. Letakkan pointer current pada node sebelum tail

        While current_node.next_node.next_node is not None:

            Current_node = current_node.next_node

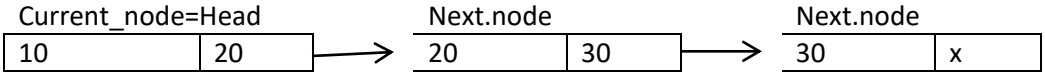
     b. Next_Node dari current diisi next_node dari tail
     c. Tail menunjuk node current

        Head

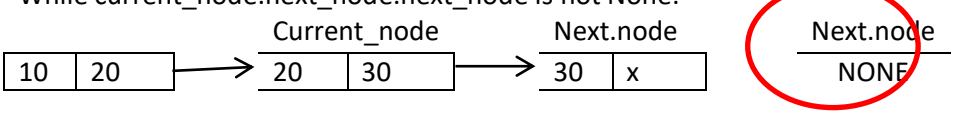


     d. Penjabaran

        o While current_node.next_node.next_node is not None:



        o While current_node.next_node.next_node is not None:



        o Current_node.next_node = self.tail.next_node