PRAKTIKUM PEMROGRAMAN I
List

Ade Sukendar

Teknik Informatika Universitas Pasundan

2024

Ready !!! Go !!!

Definisi List

- List adalah *“a finite sequence of zero or more elements”*. (Alfred & Jeffrey)
- Jika elemen *list* bertipe *T* maka tipe *list* yaitu list bertipe *T*
  - List bertipe integer
  - List bertipe bilangan pecahan
  - List bertipe struktur (misalkan Mahasiswa, Dosen, dsb)
  - List bertipe list integer (List didalam List)

Definisi List

- Penulisan elemen list dipisahkan dengan koma dan berada di dalam tanda kurung

  $(a_1, a_2, a_3, ..., a_n)$

- Model data list dapat diimplementasikan dengan array maupun *linked-list*.

Contoh List

- List bilangan prima kurang dari 20
  - (2, 3, 5, 7, 11, 13, 17, 19)
- List jumlah hari dalam setiap bulan, bukan tahun kabisat
  - (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
- Baris Teks (I, N, F, O, R, M, A, T, I, K, A, , O, Y, E, S)

Elemen List

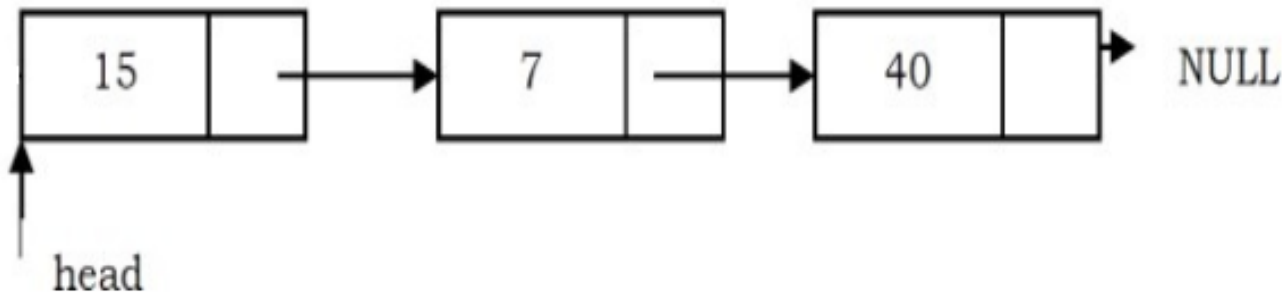
- Dari contoh list jumlah hari dalam setiap bulan adakah nilai elemen list yang sama atau duplikasi ??

Elemen List

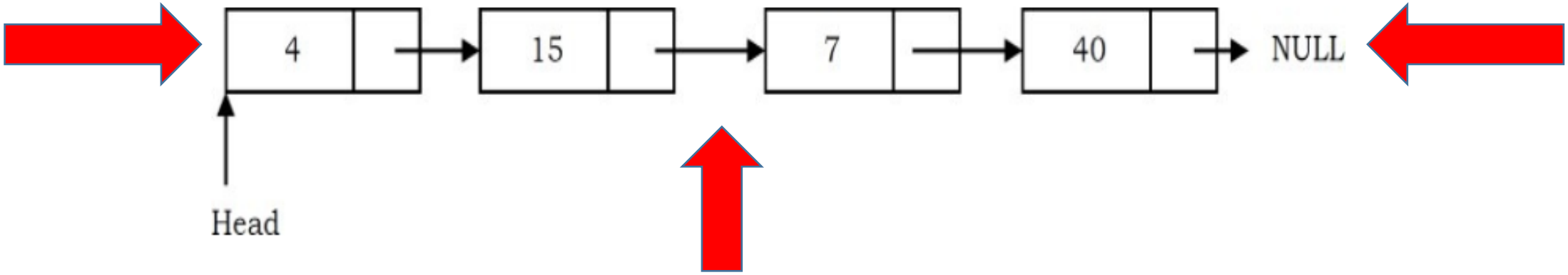
- List jumlah hari dalam setiap bulan:
- (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
- Elemen list tersebut menujukkan nilai list diperbolehkan mempunyai nilai yang sama

Elemen List

- Elemen list disebut juga *node* atau *cell*
- Node akan berisi nilai elemen (*value*) dan elemen berikutnya (*next*)
  - Nilai elemen dapat tipe primitif dan tipe bentukan user
  - Nilai elemen berikutnya berisi node berikutnya, jika berisi NULL maka tidak mempunyai relasi dengan node lain
- Contoh di bawah ini terdiri dari 3 node list



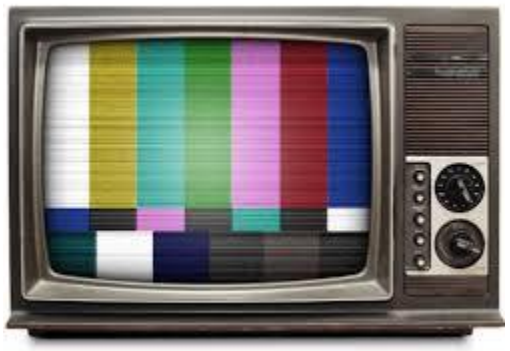
Operasi List



- Menambah (*add*) atau menghapus (*remove*)
  - elemen list di awal (*head*)
  - elemen list di tengah
  - elemen list di akhir

Operasi Tambahan List

- Mencari elemen di dalam list (*contain / find*)
- Menampilkan elemen di dalam list
- Menghitung jumlah elemen di dalam list (*count*)



Struktur Node ?

Struktur Node

- Setiap node terdiri dari **data** atau ***value*** dan ***next*** berisi ke node berikutnya
- ***Next*** menerapkan penggunaan ***by reference*** atau jika di Bahasa C menggunakan **pointer**
- Struktur Node

```
class Node {
 data: tipe data (T)
 next: Node
}
```

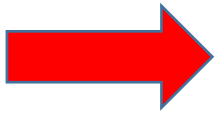
Membuat Objek Node

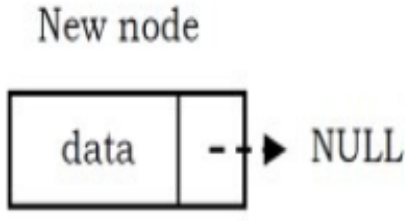
- Struktur Node perlu diimplementasikan dengan membuat sebuah kelas Node

```
class Node {

 int data;
 Node next;

 //init atribut Node
 //akses atribut Node
}
```

- Sebelum digunakan objek Node harus diinstansiasi terlebih dahulu dengan *keyword* ***new***

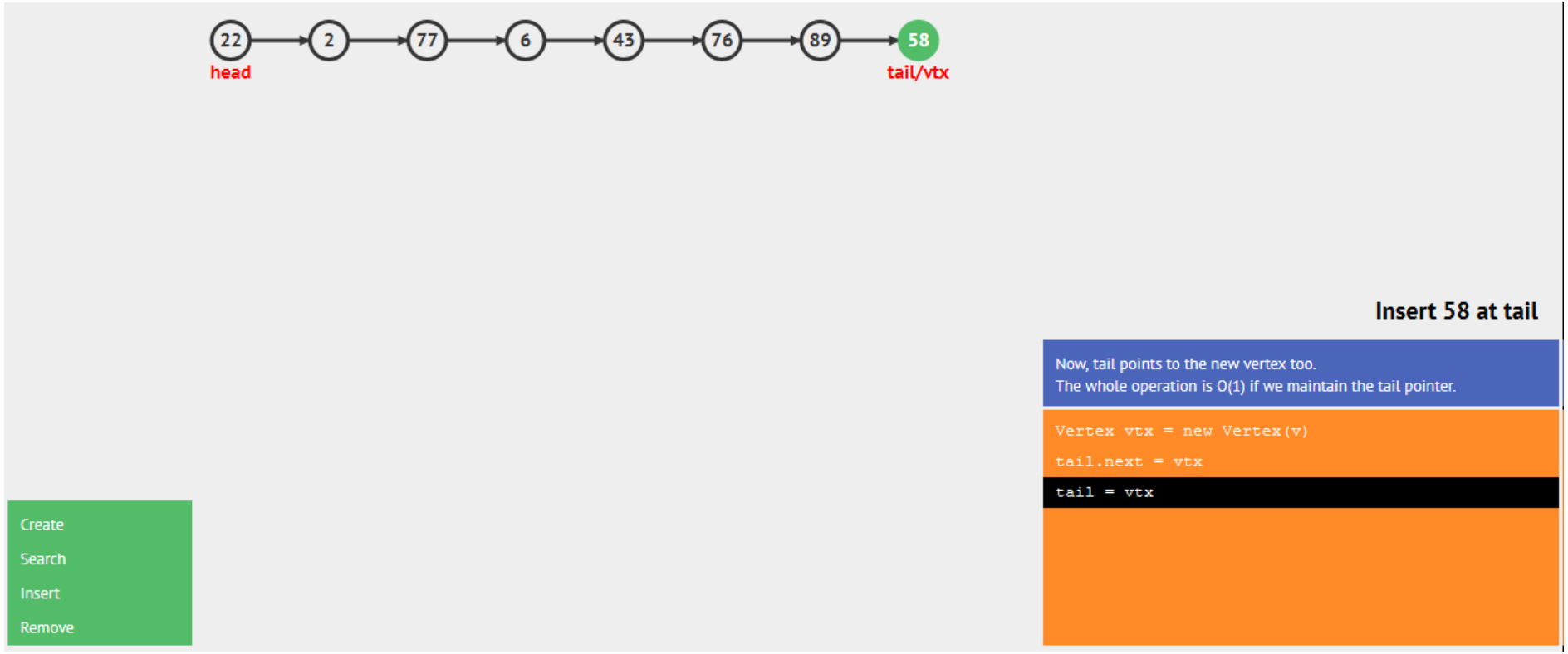
```
node = new Node(data);
```





Visualisasi Struktur Data

- Dapat di akses dari URL https://visualgo.net/en/list



Operasi List
Add Elemen di Tail

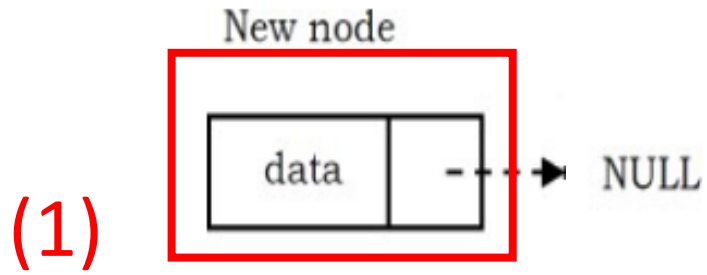
Kondisi Operasi Add Tail

- Add elemen di tail mempunyai dua kondisi yaitu
  - Jika elemen List kosong (*empty*)
  - Jika elemen List tidak kosong (*not empty*)

Proses 1: Add Tail Jika List Empty

1. Buat node baru
   a. Isi value elemen dengan nilai tertentu
   b. Isi next dengan NULL
2. Pastikan bahwa HEAD berisi NULL
3. Jadikan node baru sebagai HEAD

Proses 3: Add Tail Jika List Empty (cont..)

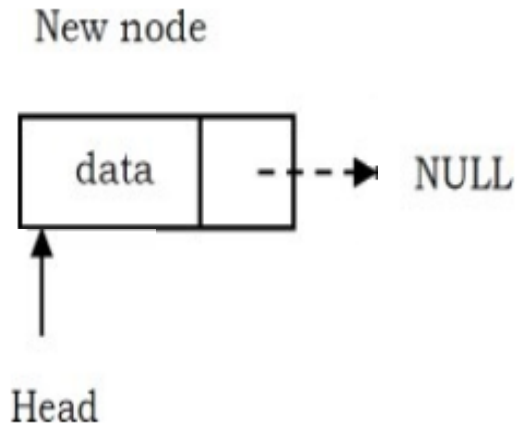


(1)

Head = NULL (2)

Node Baru menjadi Head

(3)

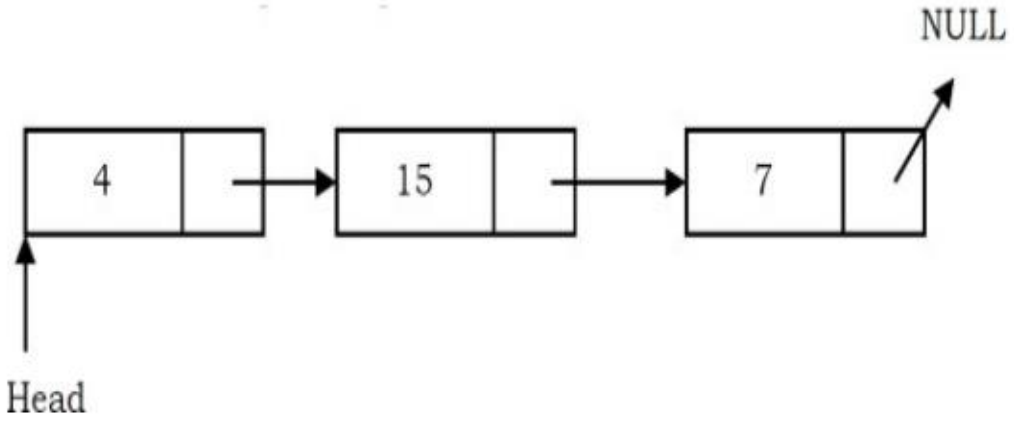


Proses 2: Add Tail Jika List Not Empty

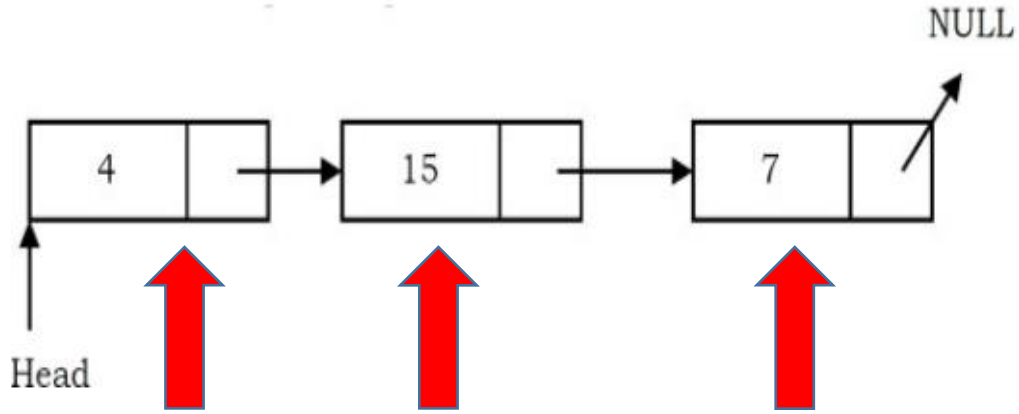
1. Buat node baru
   a. Isi value elemen dengan nilai tertentu
   b. Isi next dengan NULL
2. Pastikan bahwa HEAD tidak berisi NULL
3. Pengecekan setiap elemen node sampai menemukan nilai next node yang berisi NULL (menandakan node terakhir)
4. Next node terakhir diisi dengan node baru

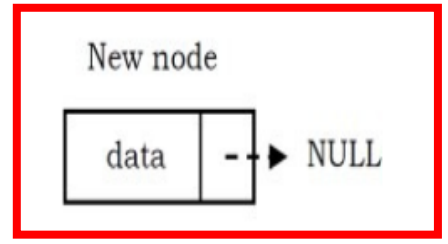
Proses 2: Add Tail Jika List Not Empty (cont..)

- Misalkan ada sebuah list yang sudah berisi elemen yaitu (4, 15, 7)
- Awal list di sebut *head, Head* menunjuk ke elemen bernilai 4
- Penanda akhir list yaitu NULL



Proses 2: Add Tail Jika List Not Empty (cont..)

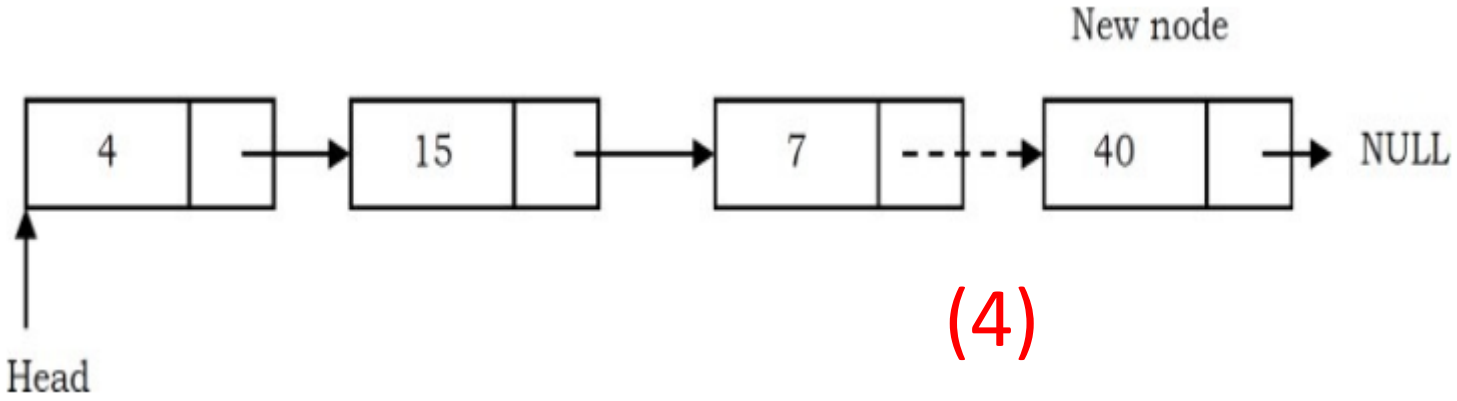




(1)

head != NULL (2)

node.next=NULL (3)



(4)

Algoritma Add Tail List

```
procedure addTail(data: integer)
deklarasi
  posNode, curNode: Node {current node}
deskripsi
  newNode ← new Node(data)
  IF (isEmpty()) THEN
    HEAD ← newNode
  ELSE
    curNode ← HEAD
    WHILE(curNode <> null) DO
      posNode ← curNode
      curNode ← curNode.next
    ENDWHILE
    posNode.next ← newNode
  ENDIF
```

Operasi List
Add Elemen di Head

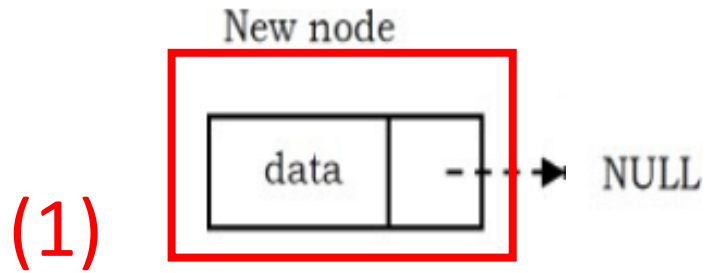
Kondisi Operasi Add Head

- Add elemen di Head mempunyai dua kondisi yaitu
  - Jika elemen List kosong (*empty*)
  - Jika elemen List tidak kosong (*not empty*)

Proses 1: Add Head Jika List Empty

1. Buat node baru
   a. Isi value elemen dengan nilai tertentu
   b. Isi next dengan NULL
2. Pastikan bahwa Head berisi NULL
3. Jadikan node baru sebagai HEAD

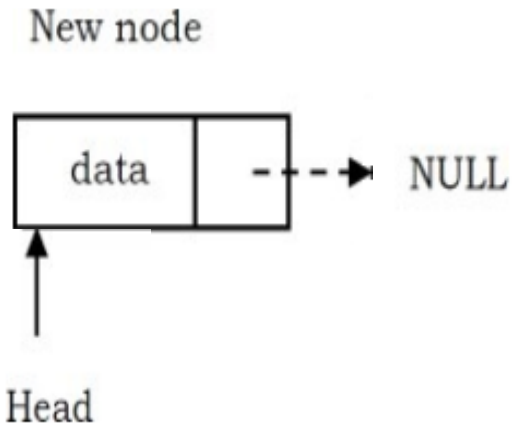
Proses 1: Add Head Jika List Empty (cont..)



(1)

Head = NULL (2)

Node Baru menjadi Head



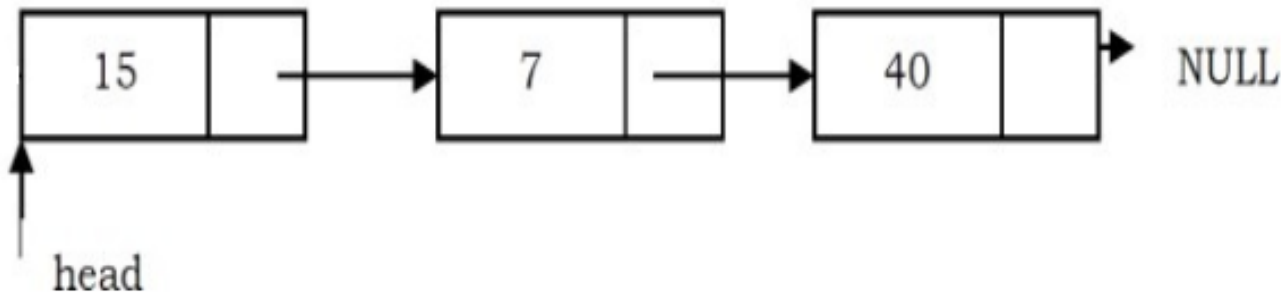
(3)

Proses 2: Add Head Jika List Not Empty

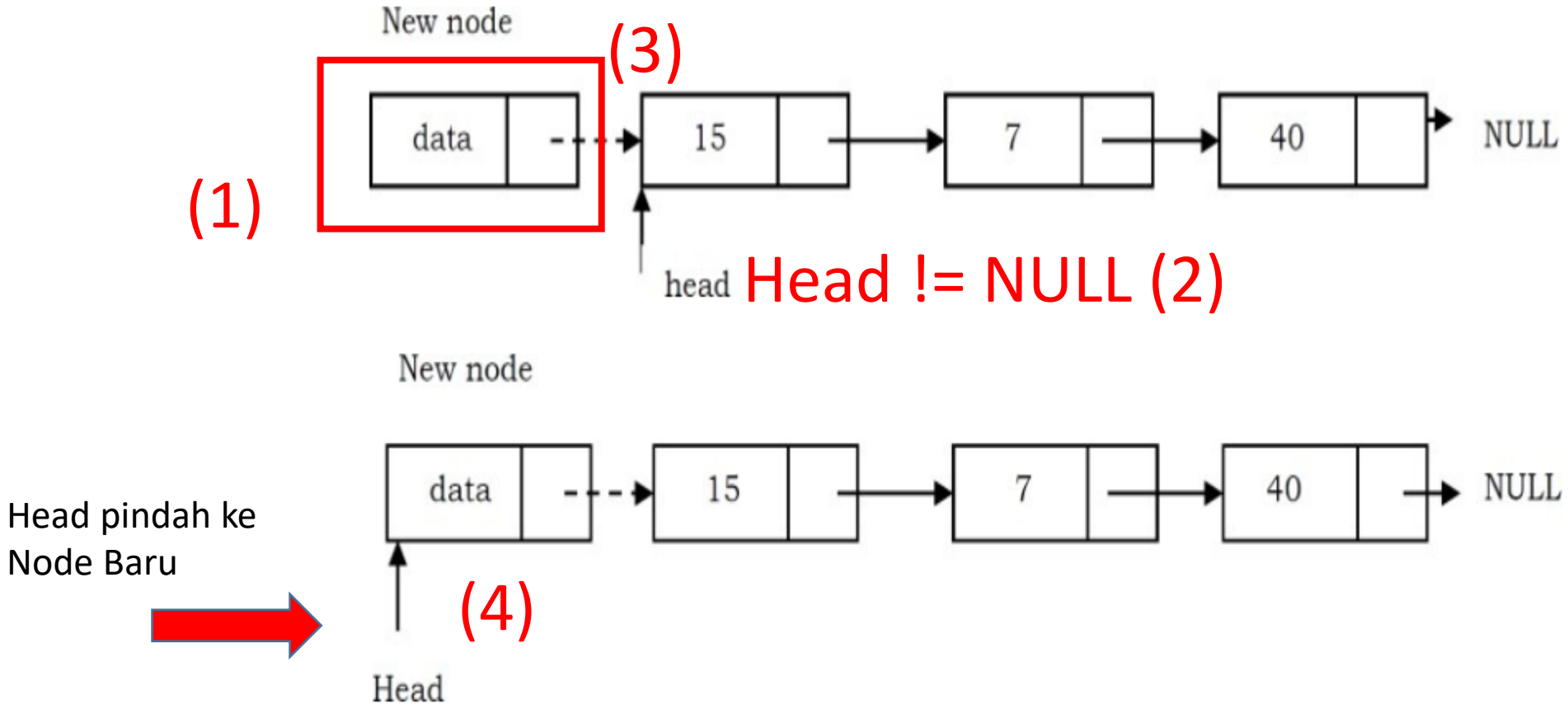
1. Buat node baru
   a. Isi value elemen dengan nilai tertentu
   b. Isi next dengan NULL
2. Pastikan bahwa HEAD tidak NULL
3. Next node baru diisi oleh head
4. Jadikan node baru sebagai HEAD

Proses 2: Add Head Jika List Not Empty (cont..)

- Misalkan ada sebuah list yang sudah berisi elemen yaitu (15, 7, 40)
- Awal list di sebut *head, Head* menunjuk ke elemen bernilai 15
- Penanda akhir list yaitu NULL



Proses 2: Add Head Jika List Not Empty (cont..)



Algoritma Add Head List

```
procedure addHead(data: integer)
deskripsi
  newNode ← new Node(data);
  IF(HEAD = null) THEN
    HEAD ← newNode
  ELSE
    newNode.next ← HEAD
    HEAD ← newNode
  ENDIF
```

Terima Kasih

Referensi

- Foundation of Computer Science – C Edition, Alfred V. Aho dan Jeffrey D. Ullman, 1994.
- Data Structures and Algorithms in Java, 2nd Edition by Robert Lafore
- Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles, Fifth Edition - Narasimha Karumanchi
- The Algorithm Design Manual - Steven S Skiena
- Algorithms (4th Edition) - Robert sedgewick