Fakultas Informatika
School of Computing
Telkom University

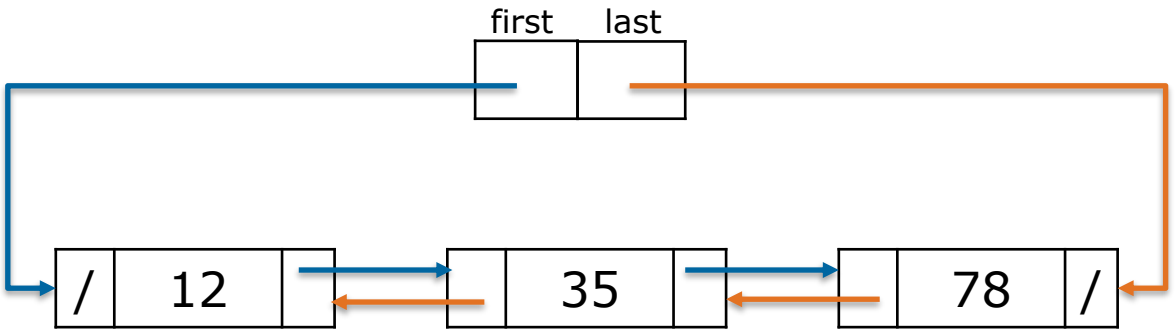
CDK2AAB4
STRUKTUR DATA

Variations of Linked List

Doubly Linked List

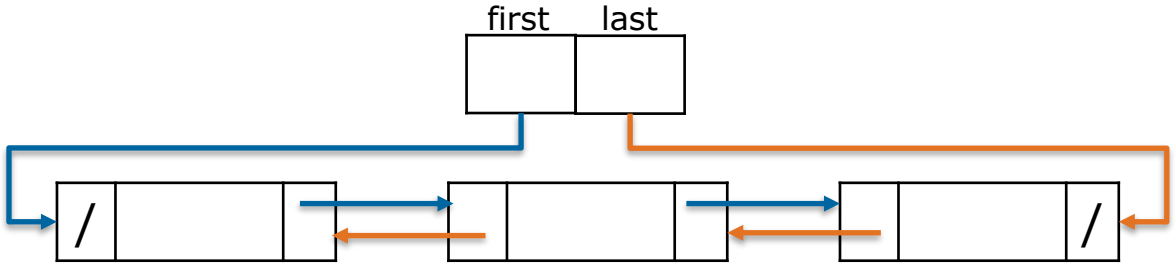
Doubly Linked List

- Linked list with 2 pointers
- Connect previous and next element

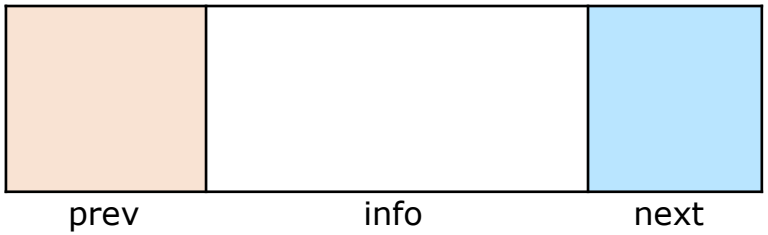


Structure

- Usually with 2 head pointer



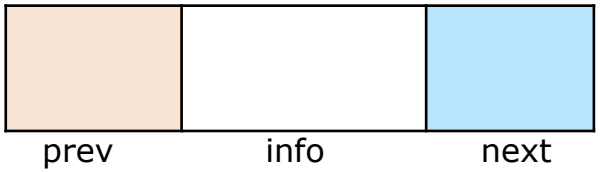
- Each element divided into 3 parts



ADT Element Doubly Linked List

```
type Infotype : integer
type Address : pointer to ElmList

type ElmList <
    info  : Infotype
    next : Address
    prev : Address
>
```

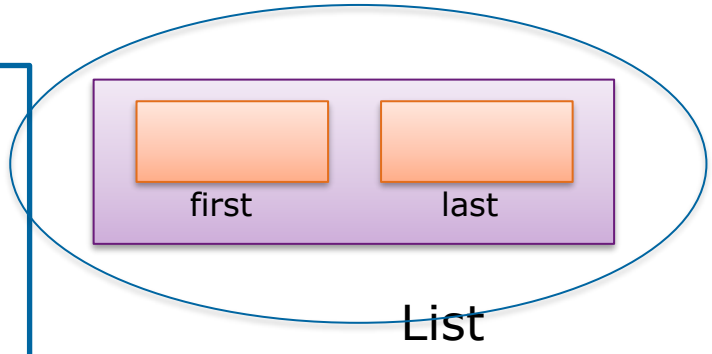


- There are 2 pointers to point the next and previous element

ADT Doubly Linked List

```
type List : <
    first : Address
    last : Address
>

Dictionary
    L : List
```



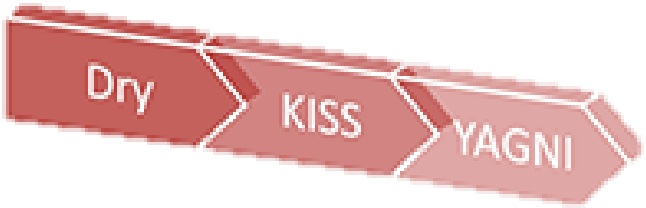
- There are 2 heads to point the first and the last element on the list

Reuse the ADT

- Try to feel the benefit of ADT
- If you have already defined ADT for Singly Linked List, modify it a little to have an ADT for Doubly Linked List
- Any modification only in specifying function (implementation)

Remember DRY Principles

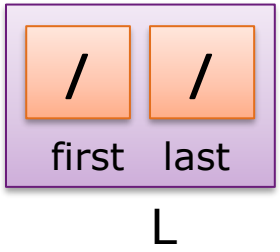
- "When you find yourself writing code that is similar or equal to something you've written before,
- "take a moment to think about what you're doing and **don't repeat yourself**."



Create New List

Algorithm

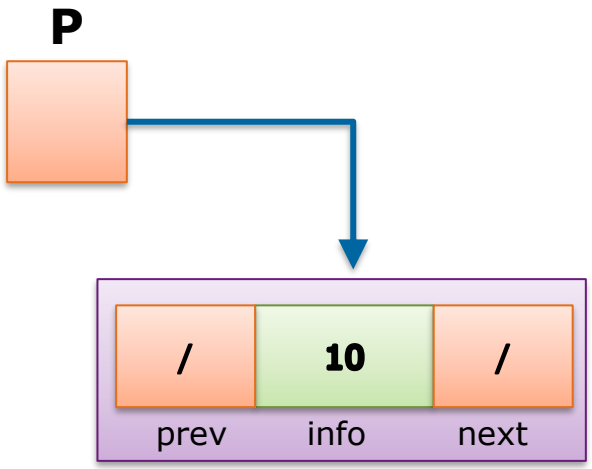
```
L.first = NIL
L.last = NIL
```



- **X.last** is a keyword to refer the last element of the list X
- On the creation of new list, there is no element, thus L.first and L.last is NIL

Creating New Element

```
Algorithm
    allocate(P)
    P→next = NIL
    P→prev = NIL
    P→info = 10
```

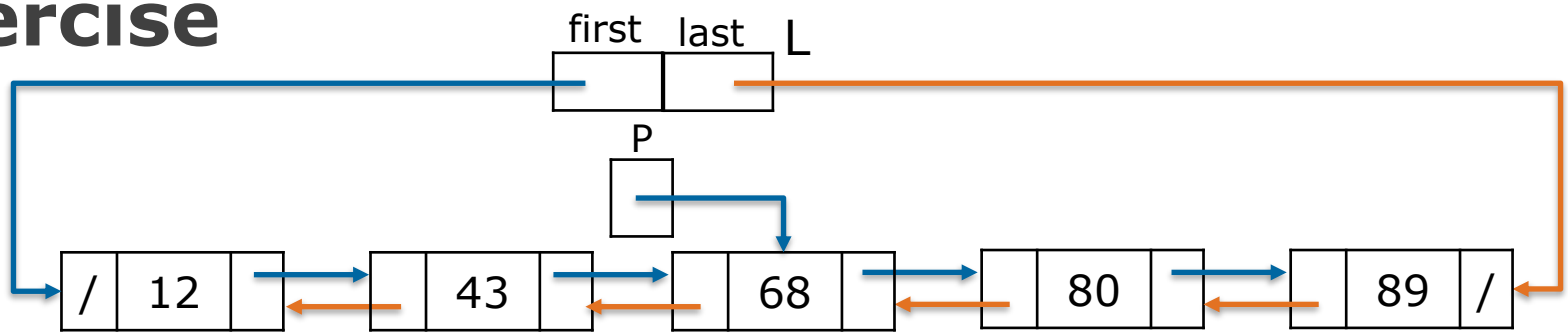


- **Y→prev** is a keyword to refer the previous element of element pointed by Y
- On the creation of new element, set next and prev element = NIL

Keywords

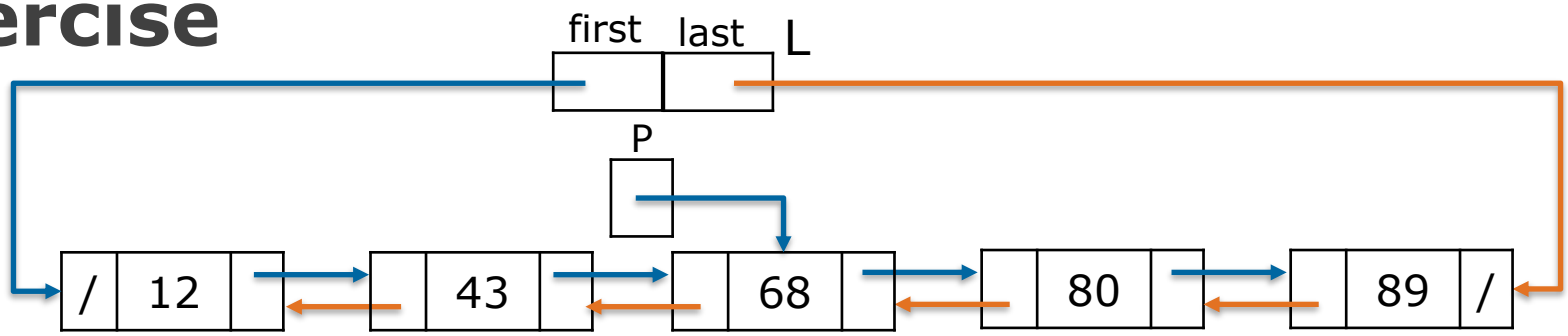
- X.first
- Y→next
- Y→info
- **X→last**
  - Select the last element of list X
- **X→prev**
  - Select the previous element of element Y

Exercise



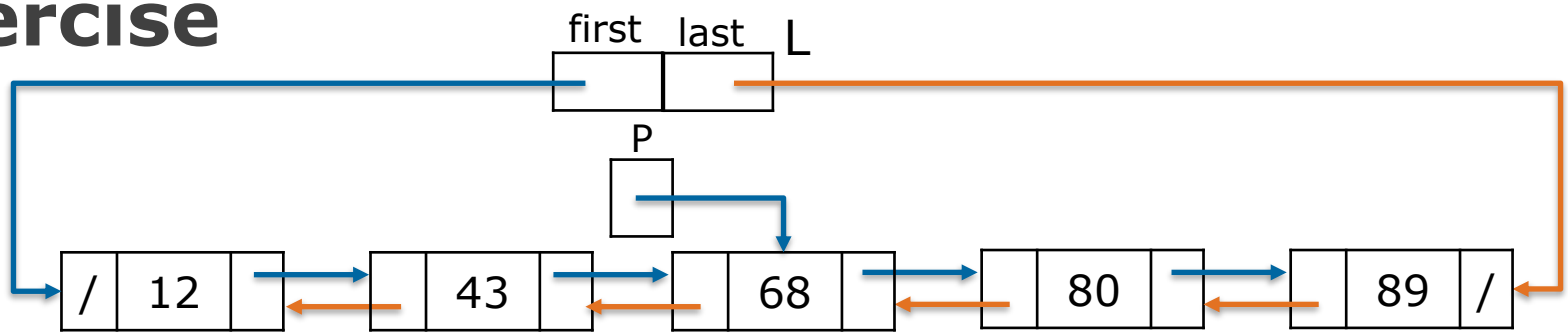
Task: Draw the Pointer
Q = (P→prev)→prev
R = (((L.last)→prev)→prev)→next
S = (((P→next)→prev)→next)→next

Exercise



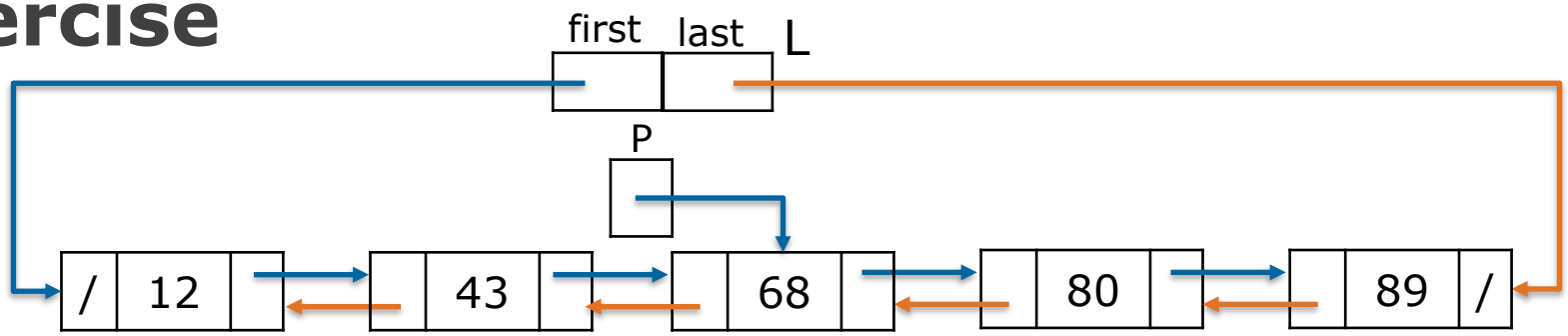
Task : How to …	output	Answer
Access info of the last element	89	
Access info of the second element of the list	43	
Access info of the fourth element of the list	80	
Copy info of 4th element to 2nd element		
Make P points 4th element		

Exercise



Task : what is the output?	Answer
((((L.last)→prev)→next)→prev)→info	
((P→next)→next)→info	
(((P→next)→prev)→prev)→info	
(((((L.first)→next)→prev)→next)→next)→info	

Exercise



Task : what is the output?	Answer
(P→next)→info + (L.last→prev)→info	
(L.first)→info – ((P→next)→next)→info	
(P→prev)→info – (((L.first)→next)→next)→info	
(((((L.first)→next)→prev)→next)→next)→info + P→info	

Question?

Inserting new Element

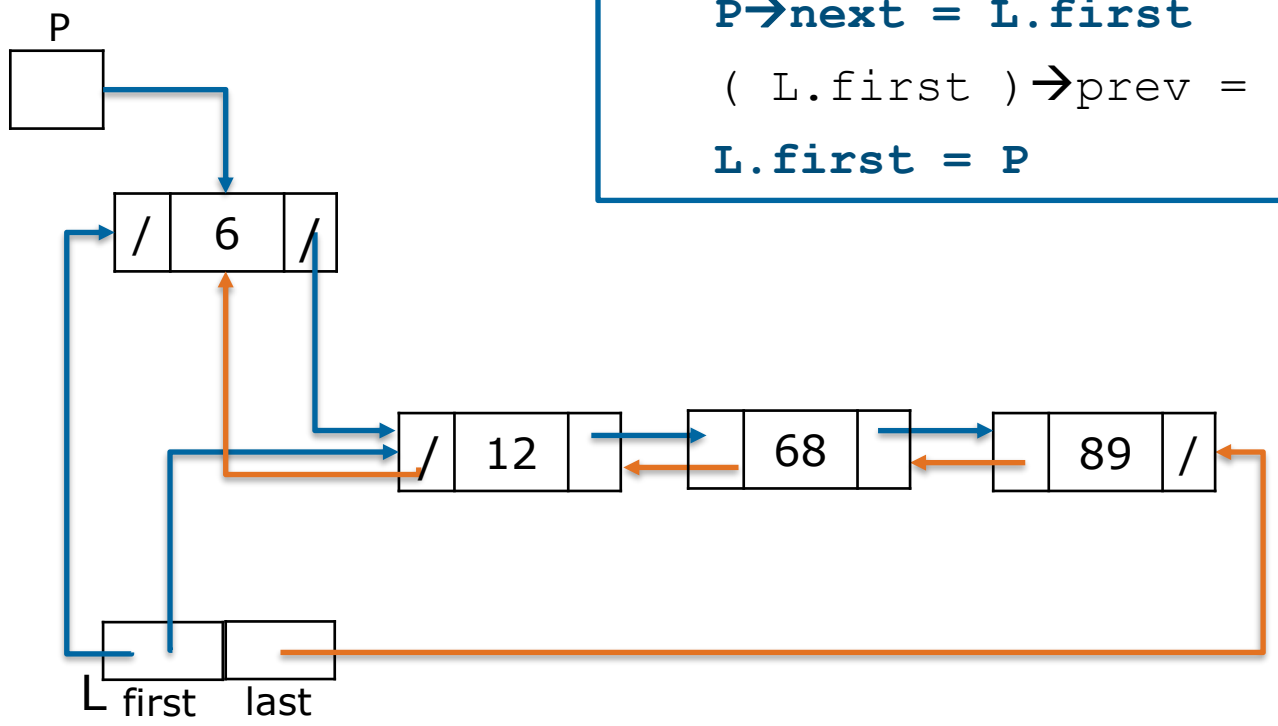
- Insert first
- Insert last
- Insert after

Insert First

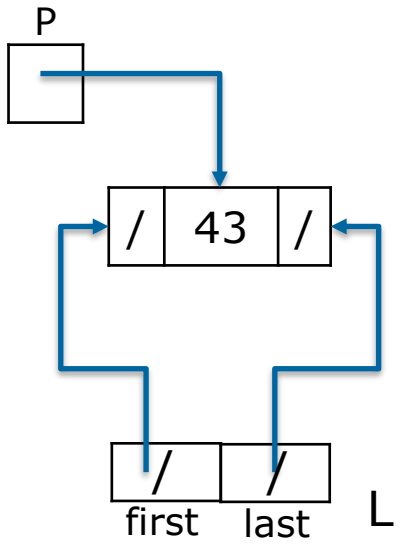
WHAT WILL HAPPEN IF THE LIST IS EMPTY?

Algorithm

```
P→next = L.first
( L.first )→prev = P
L.first = P
```



Insert First on empty list



Algorithm

```
P→next = L.first
(L.first )→prev = P
L.first = P
```

ERROR

```
// if list is empty
    L.first = P
    L.last = P
```

Insert First

Algorithm

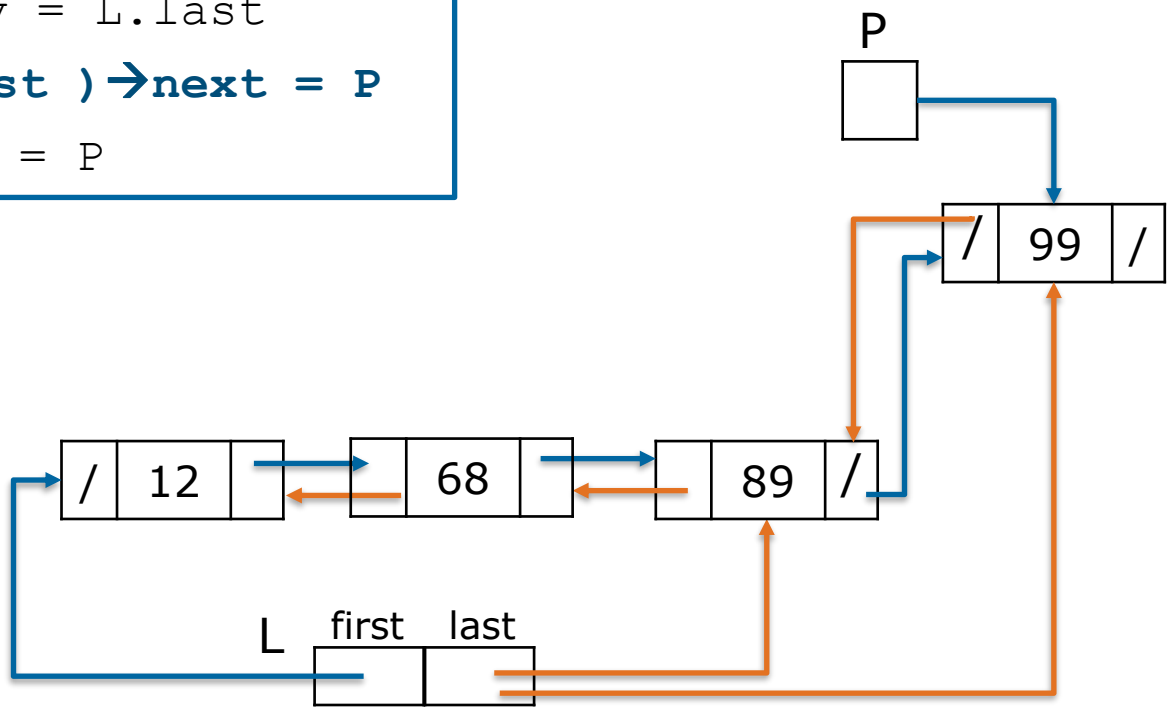
```
if L.first != NIL and L.last != NIL  then
    P.next = L.first
    (L.first )→prev = P
    L.first = P
else { List is empty }
    L.first = P
    L.last = P
endif
```

Insert Last

Algorithm

```
P→prev = L.last
( L.last )→next = P
L.last = P
```

Again, careful when the list is empty



Insert After

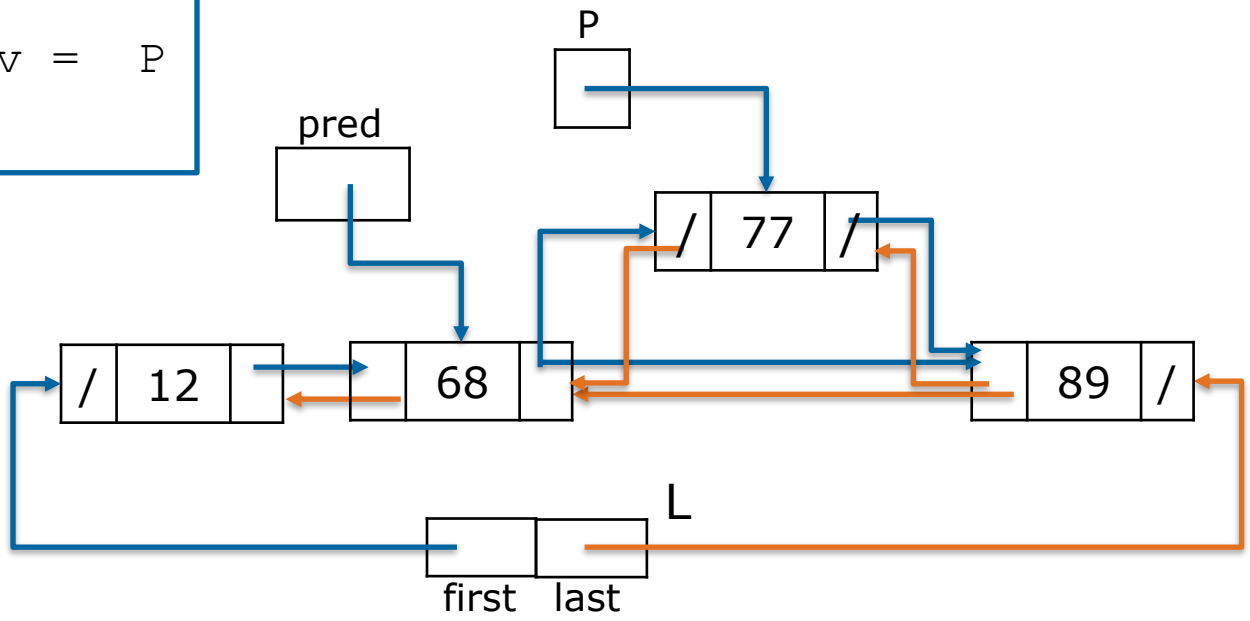
Algorithm

P→next = pred→next

P→prev = pred

(pred→next)→prev = P

pred→next = P



Deleting the Element

- Delete first
- Delete last
- Delete after

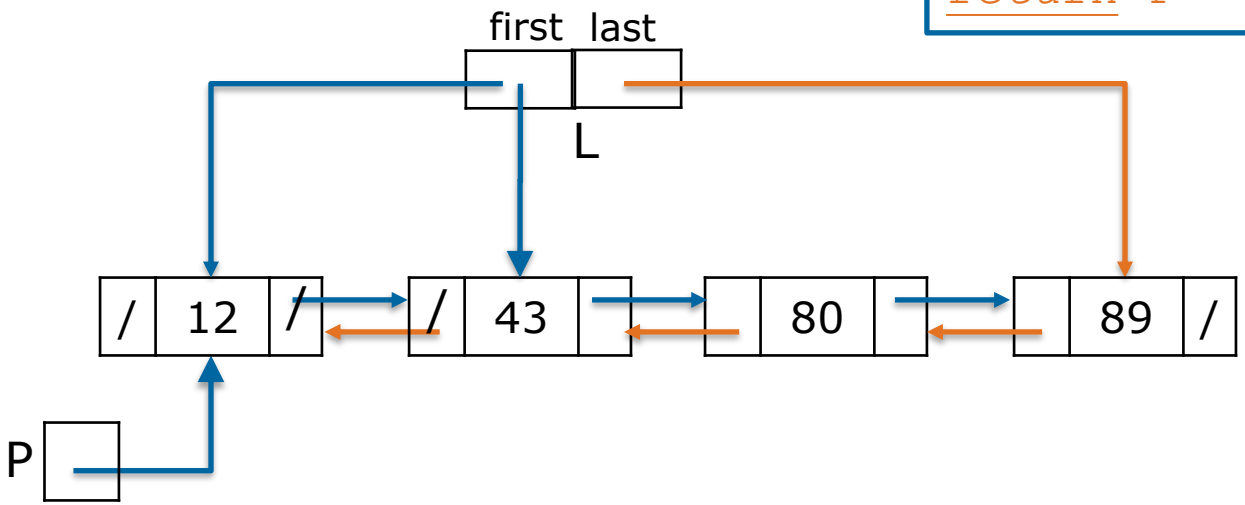
Delete First

Algorithm

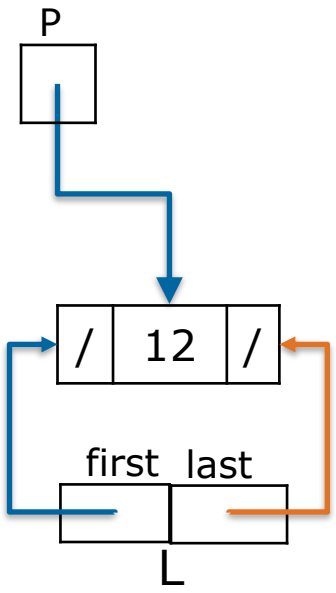
```
P = L.first
L.First = P→next
P→next = NIL
( L.first )→prev= NIL

return P
```

WHAT WILL HAPPEN IF THERE IS ONLY 1 ELEMENT INSIDE THE LIST ?



Delete First



Algorithm

P = L.first

L.first = P→next

P→next = NIL

ERROR (L.first→prev = NIL)

//if only one element

L.first = NIL

L.last = NIL

return P

Delete First

Algorithm

```
P = L.first
if L.first != L.last then
    L.first = P→next
    P→next(P) = NIL
    prev(first(L)) ← Nil
else
    L.first = NIL
    L.last = NIL
endif

return P
```

Delete Last

Algorithm

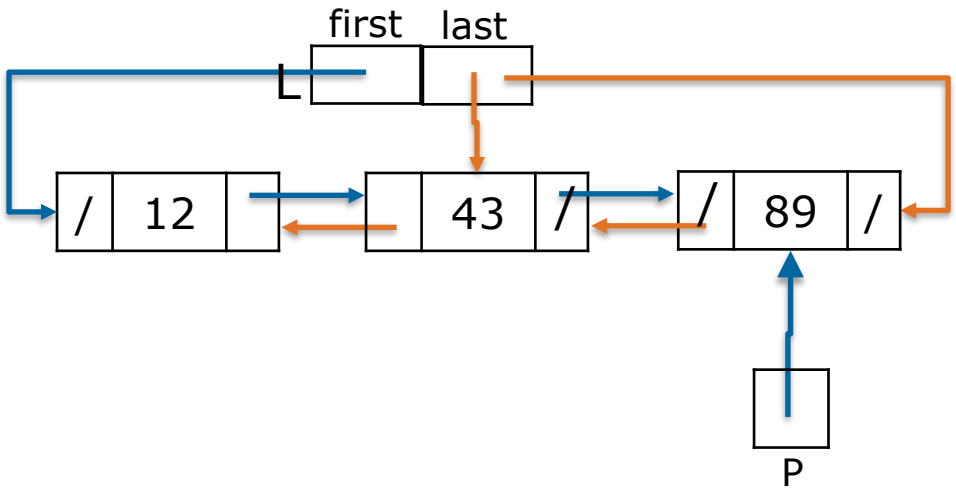
P = L.last

L.last = L.last→prev

P→prev = NIL

(L.Last)→next = NIL

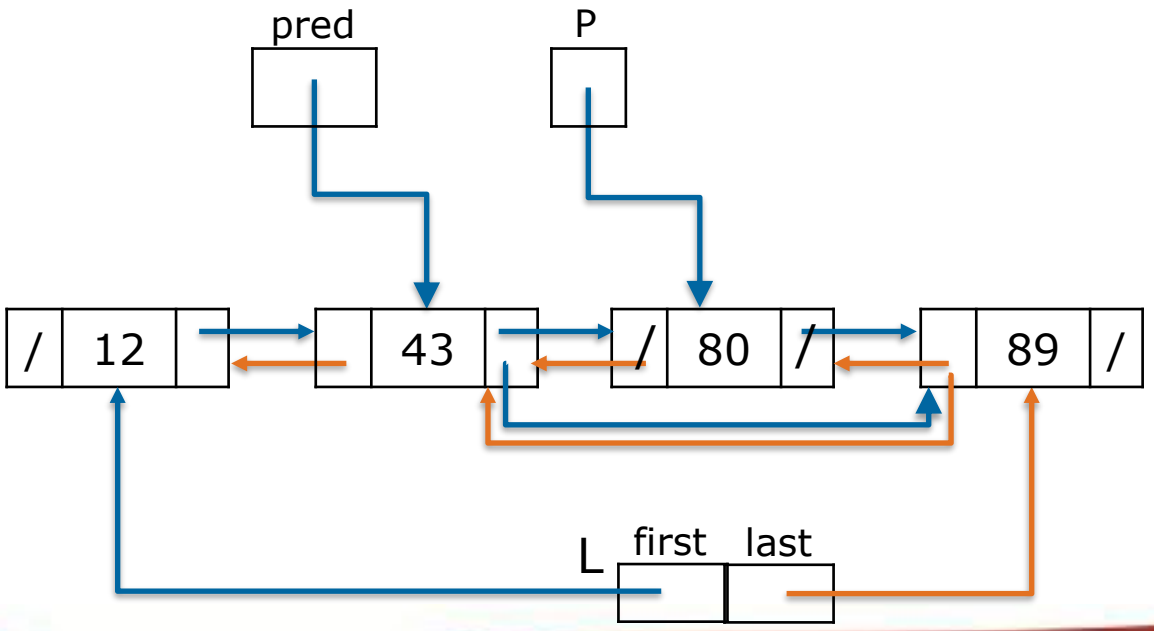
return P



Delete After

Algorithm

```
P = pred→next
Pred→next = P→next
P→next→prev = pred
P→prev= NIL
P→next = NIL
return P
```



Question?

Home Task

- Modify your previous task (Singly Linked List) into Doubly Linked List
- Write each procedure of insert and delete
- Write a function/procedure to search an element by id and output the info of the element

Note : job description should be different from the previous task

Fakultas Informatika
School of Computing
Telkom University

THANK YOU