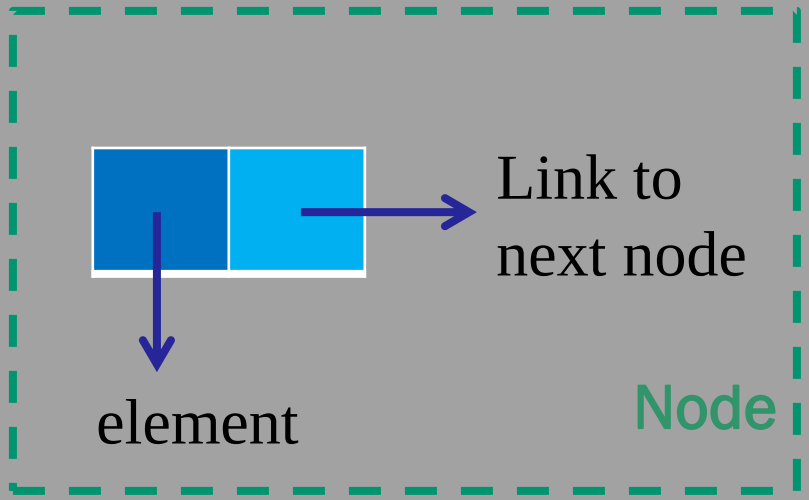
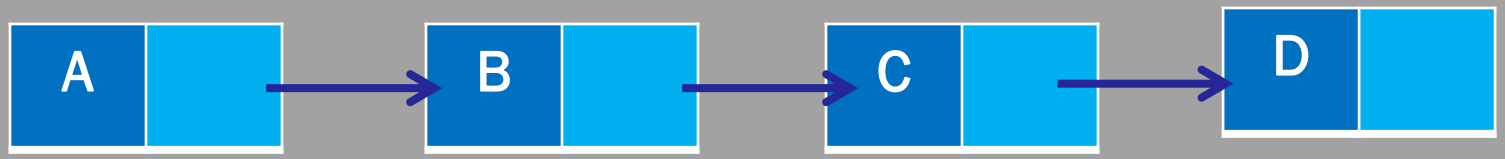
# LINKED LIST

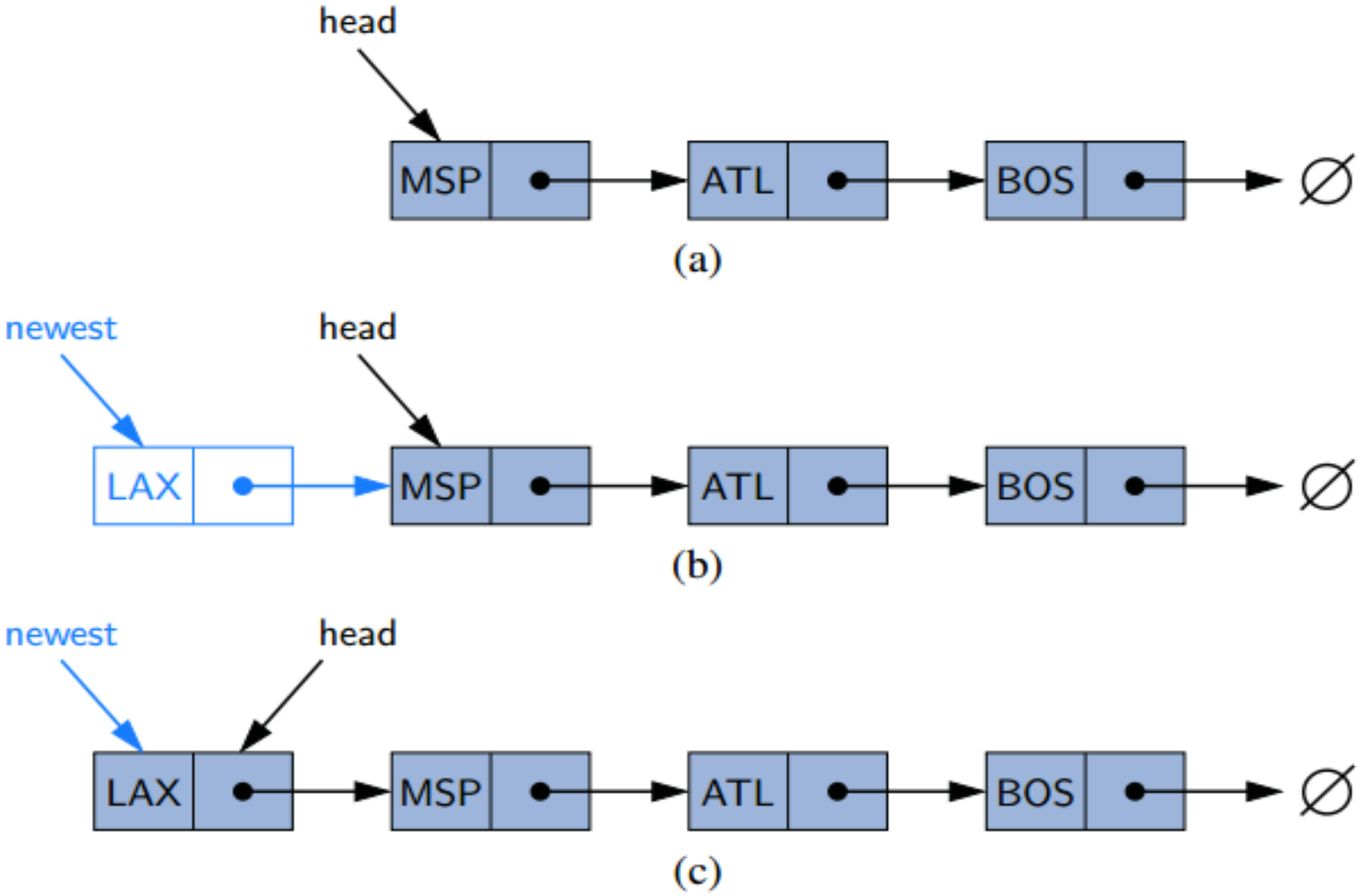
# SINGLY LINKED LISTS

- Each node stores a reference to an object that is an element of the sequence, as well as a reference to the next node of the list.
- Each node stores:
  - element
  - link to the next node



# ADD FIRST


head
MSP
ATL
BOS
∅
(a)
newest
head
LAX
MSP
ATL
BOS
∅
(b)
newest
head
LAX
MSP
ATL
BOS
∅
(c)

# ADD FIRST

*Algorithm addFirst(e):*

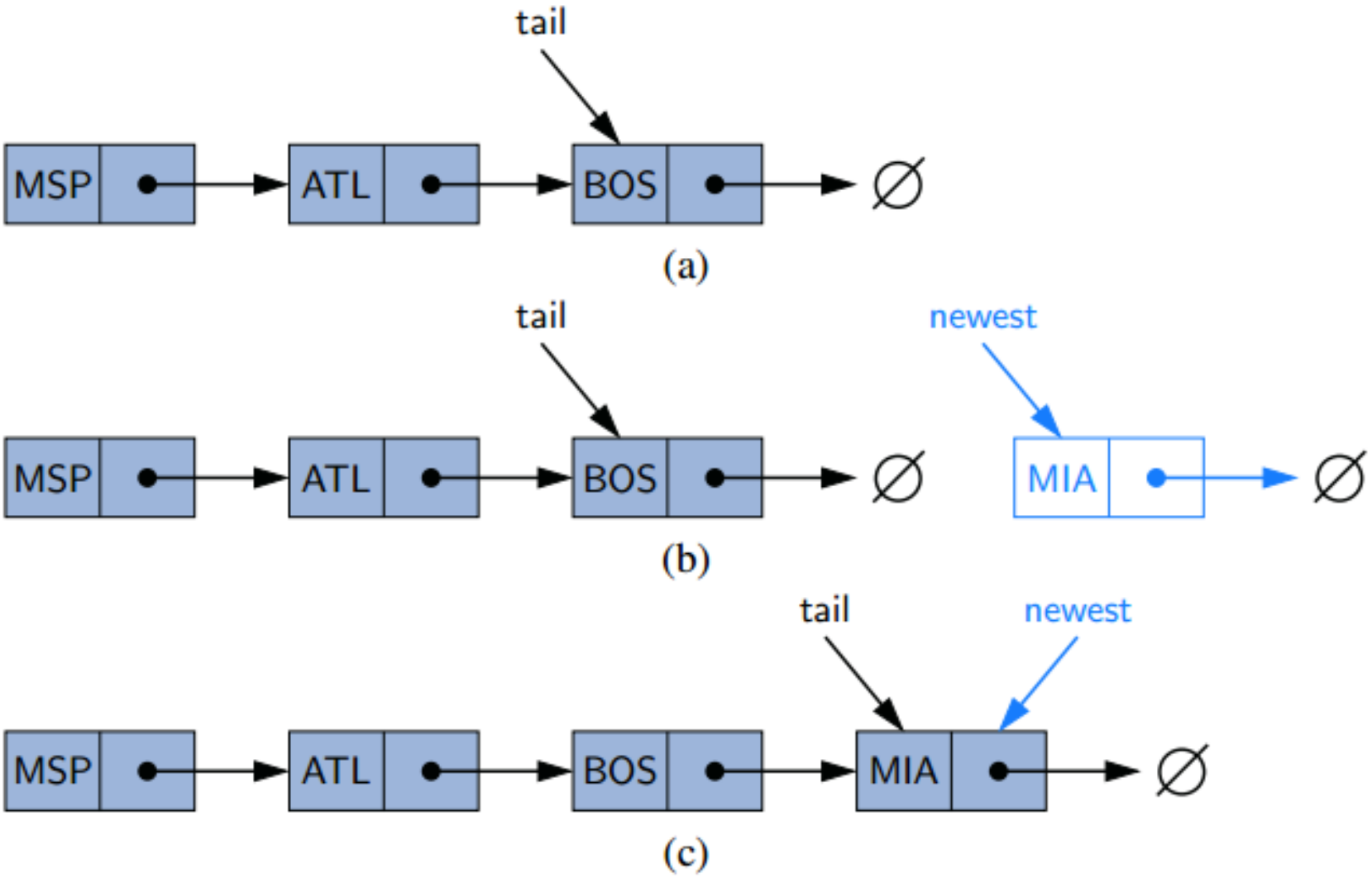
*newest=Node(e) {create new node instance storing reference to elemente}*

*newest.next=head {set new node's next to reference the old head node}*

*head=newest {set variableheadto reference the new node}*

*size=size+1 {increment the node count}*

# ADDLAST


tail
MSP
ATL
BOS
(a)
tail
newest
MSP
ATL
BOS
MIA
(b)
tail
newest
MSP
ATL
BOS
MIA
(c)

# ADDLAST

*Algorithm addLast(e):*

*newest=Node(e) {create new node instance storing reference to elemente}*

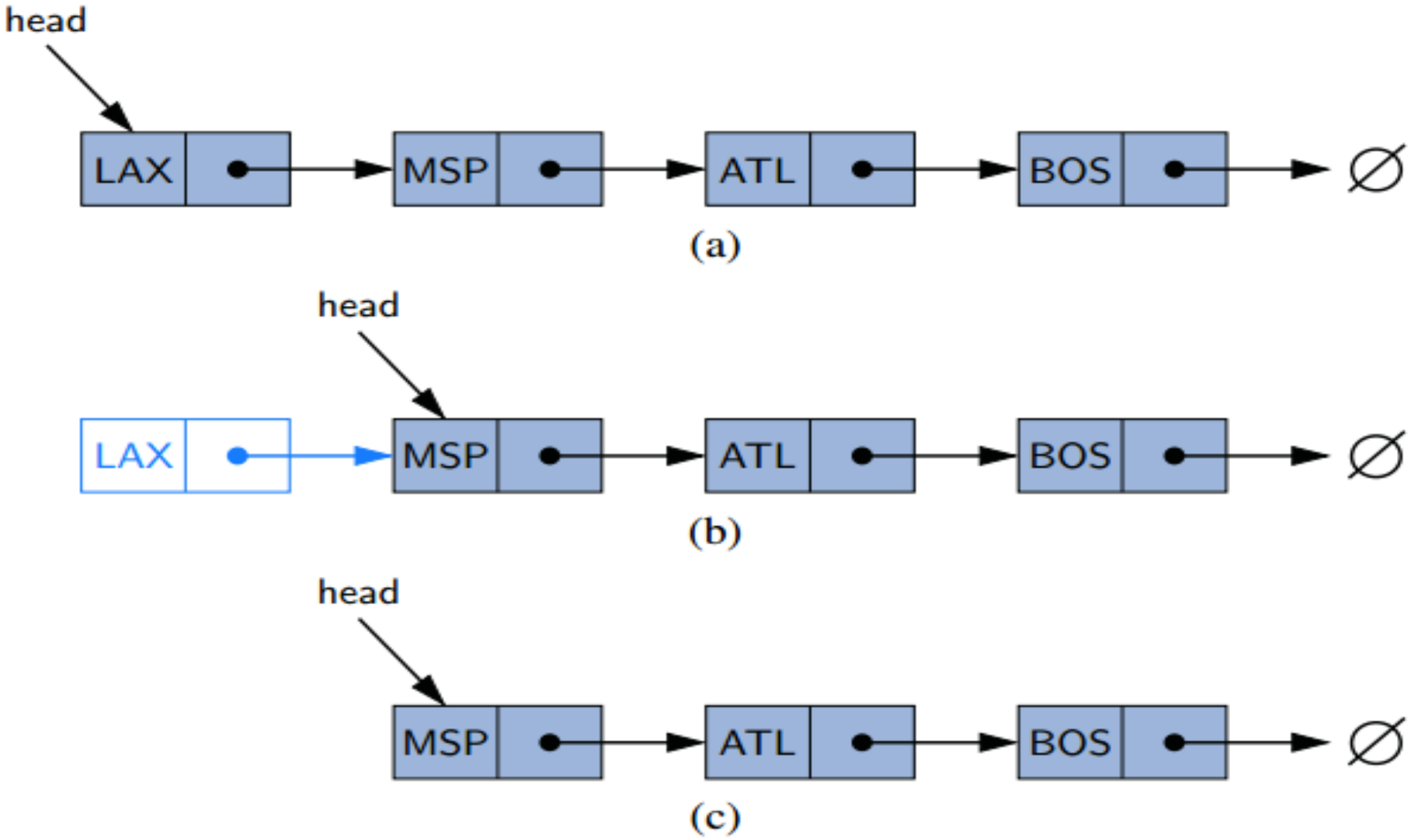
*newest.next=null {set new node's next to reference thenullobject}*

*tail.next=newest {make old tail node point to new node}*

*tail=newest {set variabletailto reference the new node}*

*size=size+1 {increment the node count}*

# REMOVE



# REMOVE

*Algorithm RemoveFirst( ):*

*If (head==null) then*

*the list is empty.*

*head=head.next {make head point to next node (or null)}*

*size=size−1 {decrement the node count}*

# IMPLEMENTING A SINGLY LINKED LIST CLASS

size( ): Returns the number of elements in the list.

isEmpty( ): Returns **true** if the list is empty, and **false** otherwise.

first( ): Returns (but does not remove) the first element in the list.

last( ): Returns (but does not remove) the last element in the list.

addFirst(*e*): Adds a new element to the front of the list.

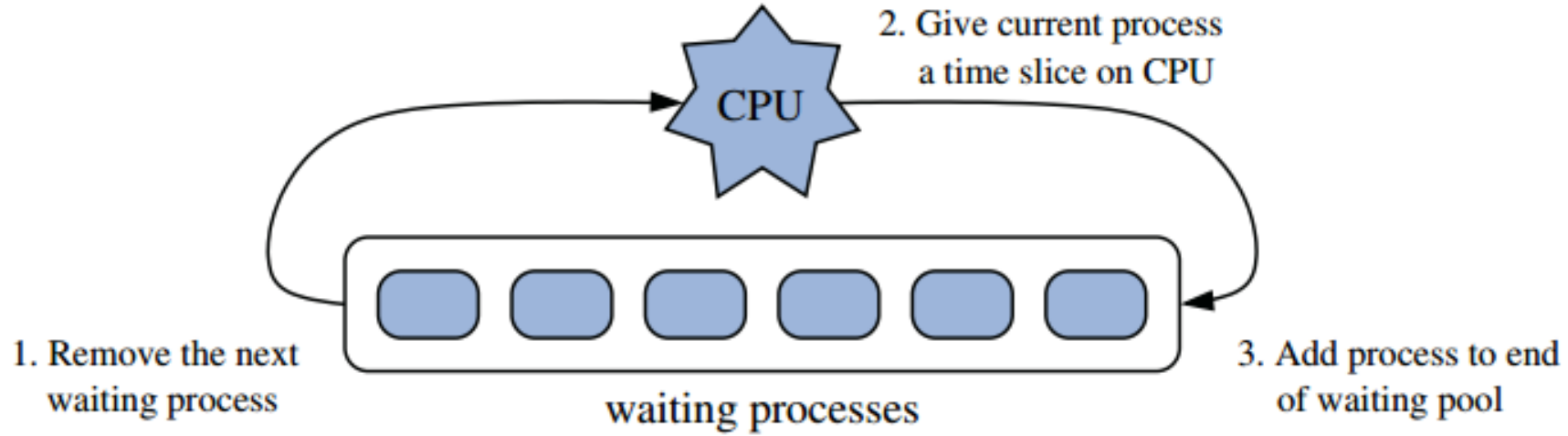
addLast(*e*): Adds a new element to the end of the list.

removeFirst( ): Removes and returns the first element of the list.

# IMPLEMENTING A SINGLY LINKED LIST CLASS

- Read and do exercise in pg127

# CIRCULARLY LINKED LISTS



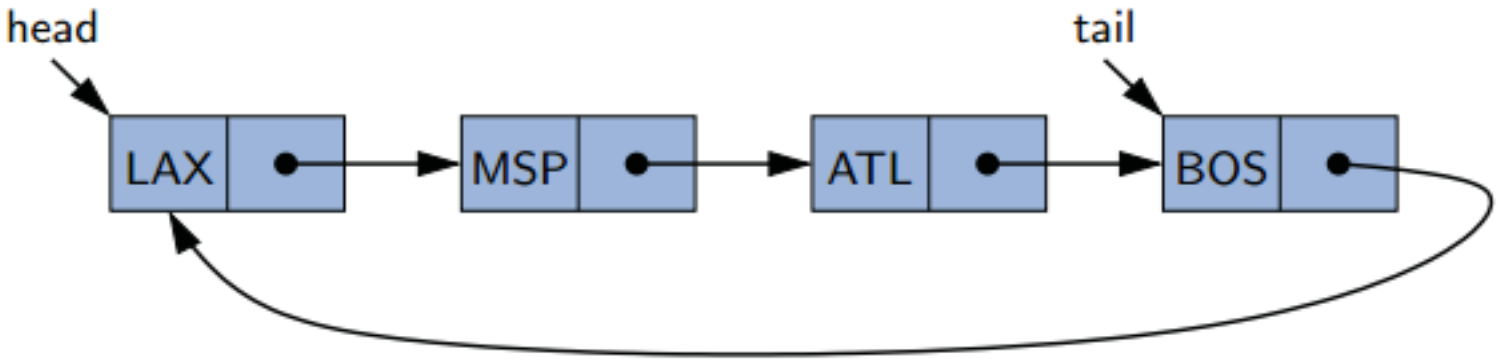
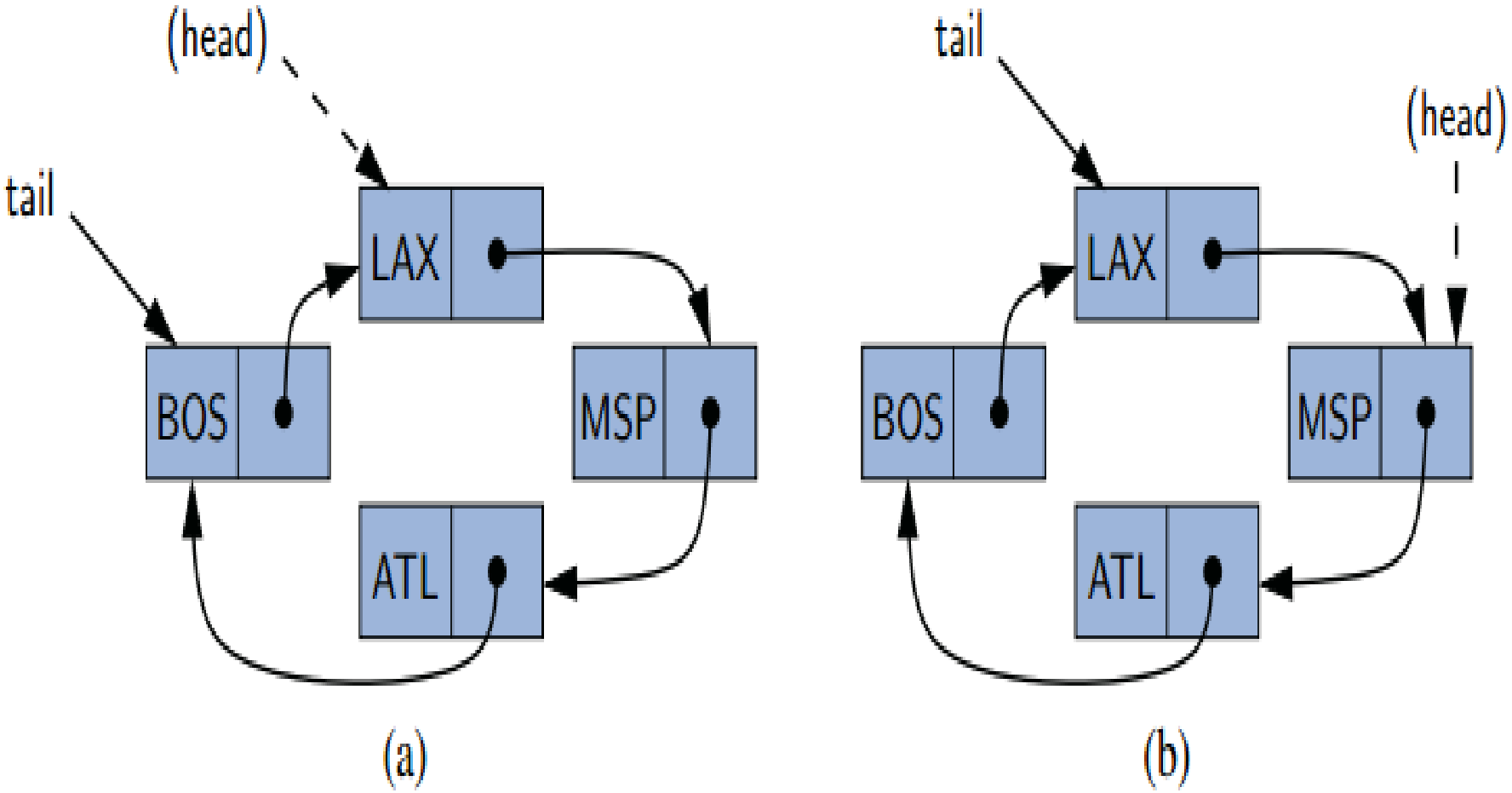
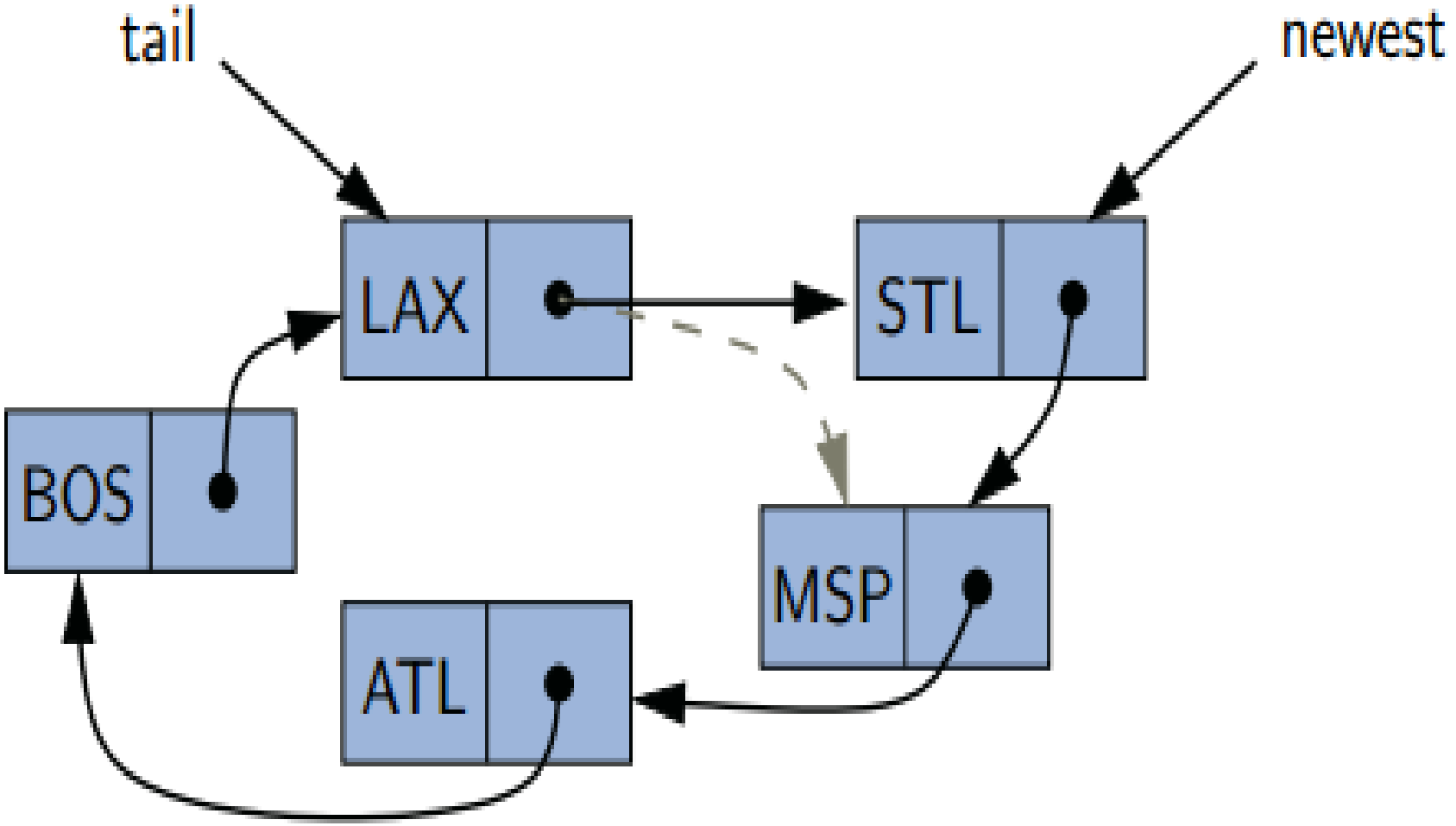




**Figure 3.16:** Example of a singly linked list with circular structure.

# ROTATE



# ADD FIRST

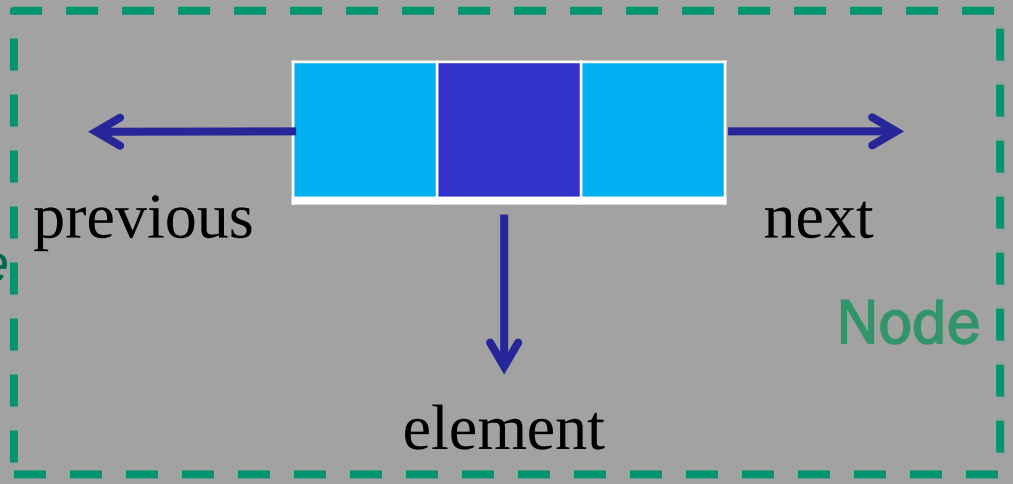


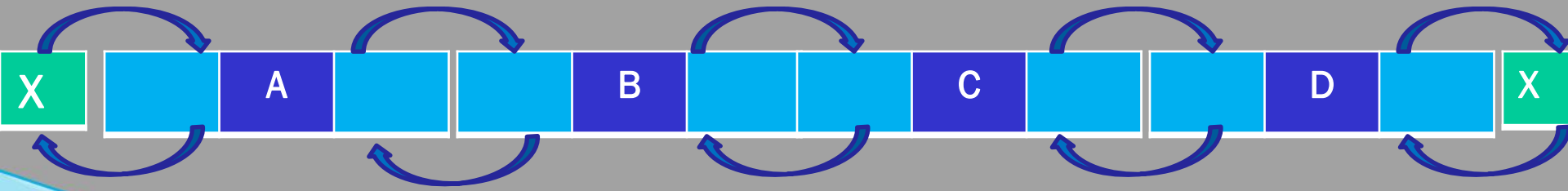
# IMPLEMENTING A CIRCULARLY LINKED LIST CLASS

*Read and do exercise in pg131*

# DOUBLY LINKED LISTS

- Each node stores a reference to an object that is an element of the sequence, as well as TWO references: the next node of the list, previous node of the list.
- Each node stores:
  - *element*
  - *link to the previous node*
  - *link to the next node*

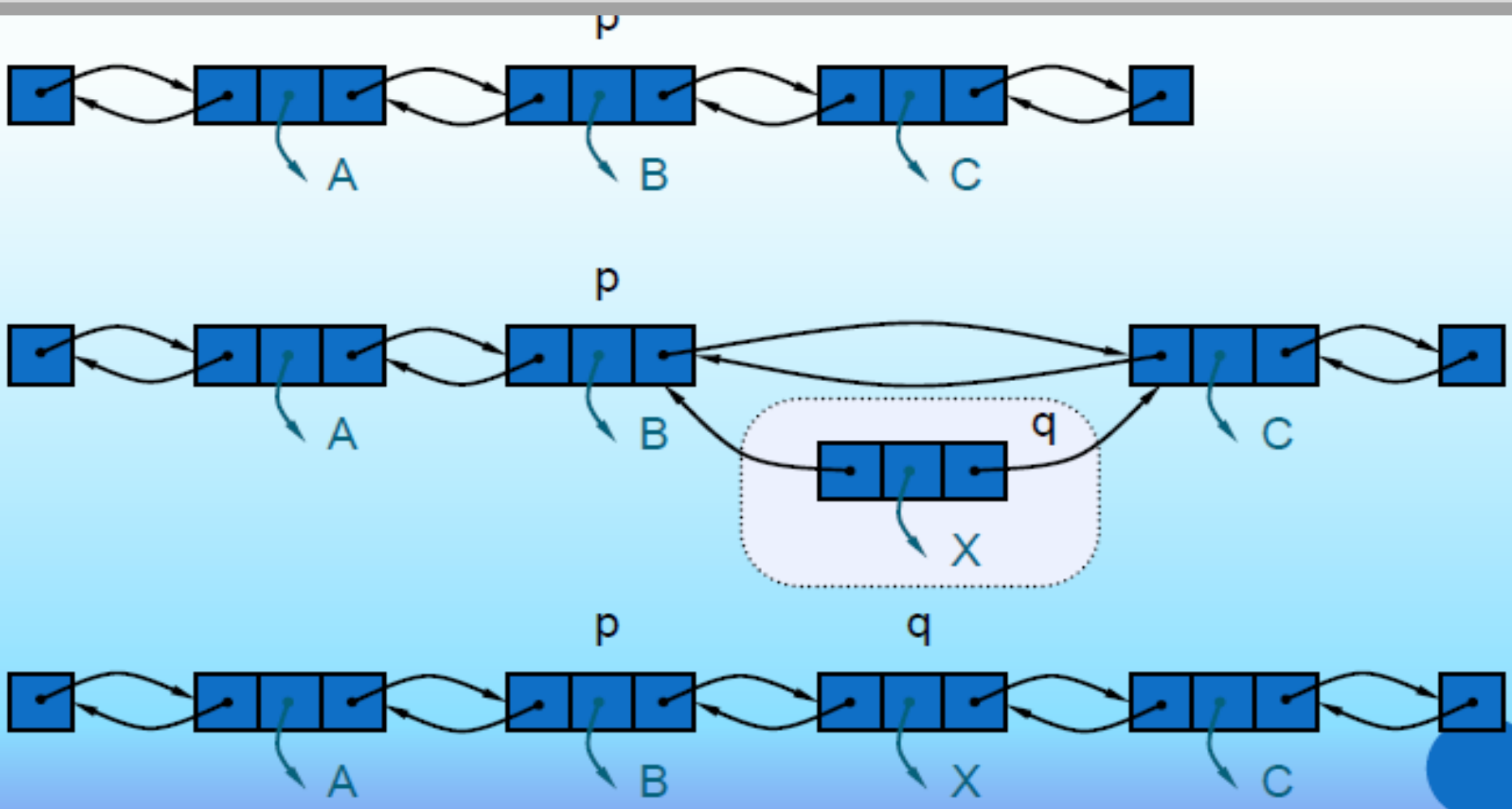


# DOUBLY LINKED LIST

- *A doubly linked list is often more convenient!*
- *Nodes store:*
  - *element*
  - *link to the previous node*
  - *link to the next node*

# INSERTION

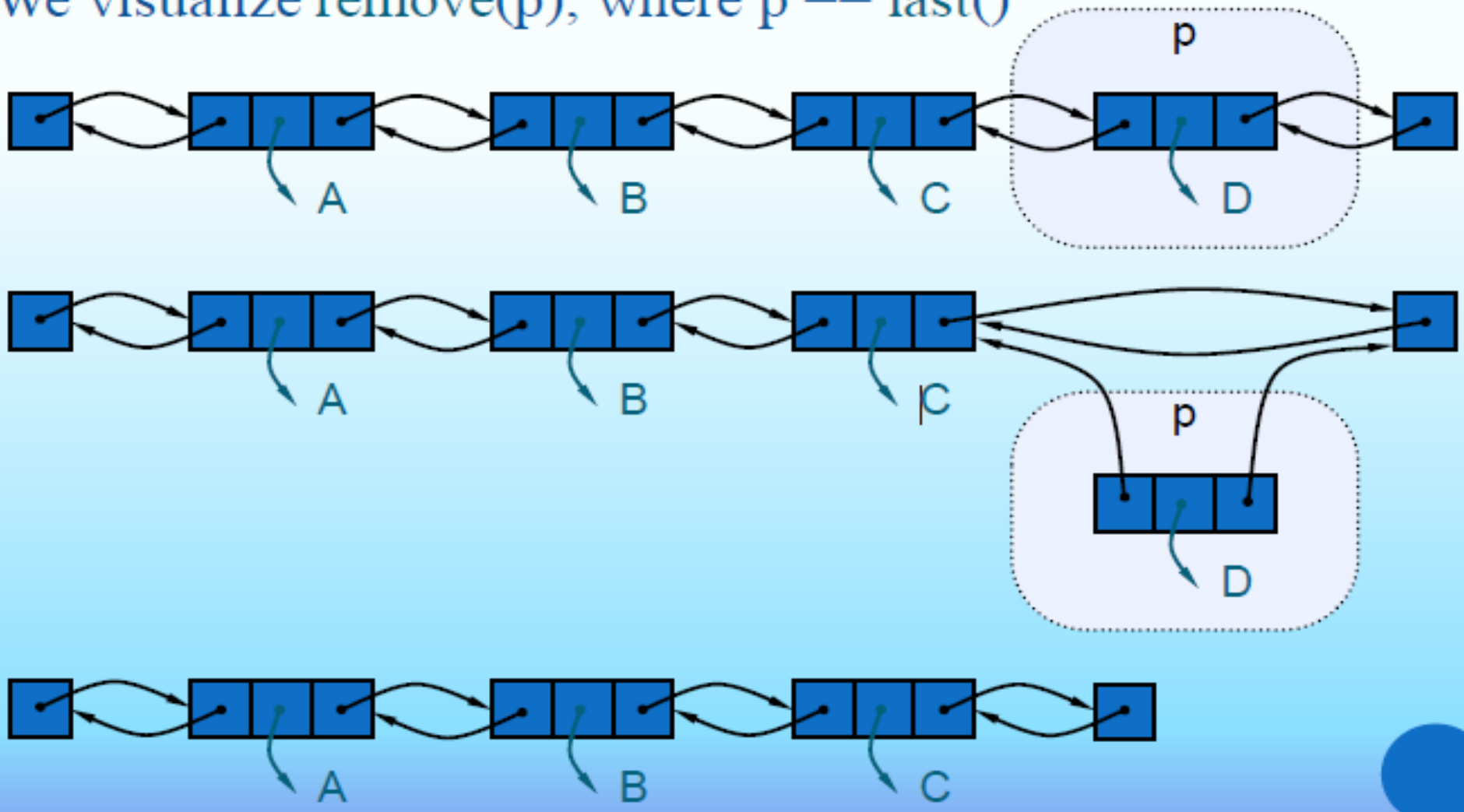


# INSERTION ALGORITHM

*Algorithm insertAfter(p,e):*
*Create a new node v*
*v.setElement(e)*
*v.setPrev(p) {link v to its predecessor}*
*v.setNext(p.getNext()) {link v to its successor}*
*(p.getNext()).setPrev(v) {link p's old successor to v}*
*p.setNext(v) {link p to its new successor, v}*
*return v {the position for the element e}*

# DELETE

We visualize remove(p), where p == last()



# DELETION ALGORITHM

**Algorithm** remove(*p*):
*t* = *p.*element {a temporary variable to hold the return value}
(*p.*getPrev()).setNext(*p.*getNext()) {linking out *p*}
(*p.*getNext()).setPrev(*p.*getPrev())
*p.*setPrev(**null**) {invalidating the position *p*}
*p.*setNext(**null**)
**return** *t*

# IMPLEMENTING A DOUBLY LINKED LIST

| | |
|---|---|
| size( ): | Returns the number of elements in the list. |
| isEmpty( ): | Returns **true** if the list is empty, and **false** otherwise. |
| first( ): | Returns (but does not remove) the first element in the list. |
| last( ): | Returns (but does not remove) the last element in the list. |
| addFirst(*e*): | Adds a new element to the front of the list. |
| addLast(*e*): | Adds a new element to the end of the list. |
| removeFirst( ): | Removes and returns the first element of the list. |
| removeLast( ): | Removes and returns the last element of the list. |

# LINKEDLIST IN JAVA COLLECTION FRAMEWORK

# CONSTRUCTORS

- *LinkedList() : constructs an empty linked list.*
- *LinkedList(Collection<? extends E> elements) : constructs a linked list and adds all elements from a collection.*

# MAIN METHODS

- void addFirst(E element) : add an element to the beginning of the list.
- void addLast(E element) : add an element to the end of the list.
- E getFirst() : return the element at the beginning of the list
- E getLast() : return the element at the end of the list.

# SMALL EXERCISE

1) *CHECK SOME METHODS OF LINKED LIST IN JAVA COLLECTION FRAME WORK*
2) *HOW TO COMBINE 2 LINKED LISTS TOGETHER ?*
3) *HOW TO REVERSE LINKED LIST?*

# MANAGE SCORE OF CLASS

- *Pupil has: id, full name, average mark*
- *Class has: id, name, year and a list of sorted pupil by average mark.*
- *Requirements:*
- Sorting pupil by average mark(Hint: using addFirst(), addLast() in linked list to writing sort())
- Getting top pupils in class
- Getting bottom pupils in class
- Finding average of pupil if known name of pupil
- Getting average mark of class
- Remove all pupil has mark less than average mark of class

# DIGITAL LOTTE

## WINNING RESULTS MEGA 6/45

Draw No. #00045 | Draw Date 30/10/2016

Trực tiếp trên VTC 7 -Todaytv vào 18h00 thứ 4 - thứ 6 - CN hàng tuần

07 12 25 28 40 44

Jackpot amount

52.703.398.000 dong

| Prize | Matching | Number of prize | Prize amount (dong) |
|---|---|---|---|
| Jackpot | ○○○○○○ | 0 | 52.703.398.000 |
| 1st Prize | ○○○○○ | 72 | 10.000.000 |
| 2nd Prize | ○○○○ | 3160 | 300.000 |
| 3rd Prize | ○○○ | 53369 | 30.000 |

See past draws ›


Vietlott

WARNING

FOR CUSTOMERS

# DIGITAL LOTTE

- *Làm hệ thống quản lý  các điểm bán vé, số vé phát hành tại điểm, quản lý vé trúng thưởng, tiền tích lũy qua các kỳ vé*

Mega 6/45

| Prize level | Matching | Value (VNĐ) | Projected payout rate |
|---|---|---|---|
| Jackpot | ●●●●●● | Min 12 billion (accumulated) | 41.31% |
| 1st | ●●●●● | 10.000.000 | 2.87% |
| 2nd | ●●●● | 300.000 | 4.09% |
| 3rd | ●●● | 30.000 | 6.73% |
| **Total** | | | **55.00%** |

Note: ● is number matching with draw result