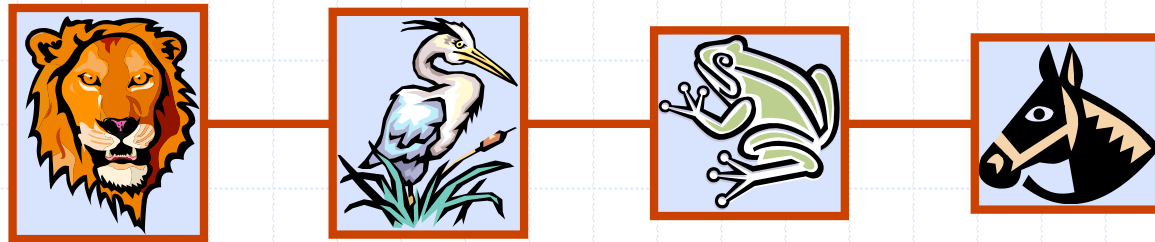
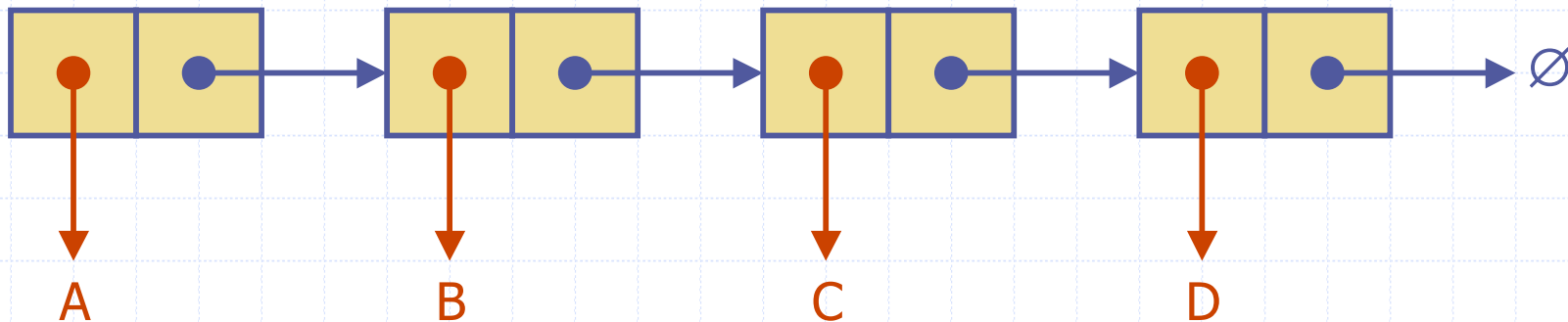
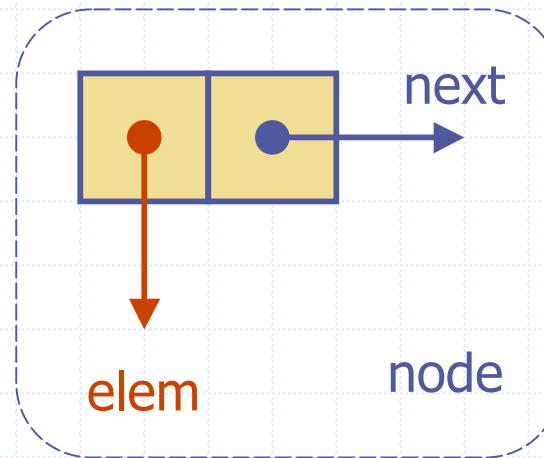


# Linked Lists



# Singly Linked List (§ 4.4.1)

- ◆ A singly linked list is a concrete data structure consisting of a sequence of nodes
- ◆ Each node stores
  - element
  - link to the next node

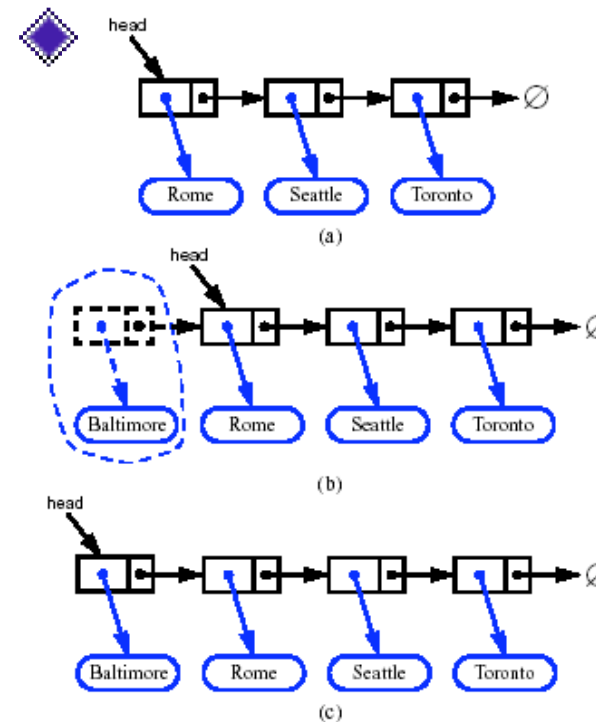


# Linked List node in Java

```
public class ListNode {  
    Object elem;  
    ListNode next;  
    public ListNode (ListNode next, Object elem)  
        {this.next = next; this.elem = elem; }  
    public Object getElem() {return elem; }  
    public void setElem(Object elem) {this.elem = elem; }  
    public ListNode getNext() {return next; }  
    public void setNext(ListNode next) {this.next = next; }  
}
```

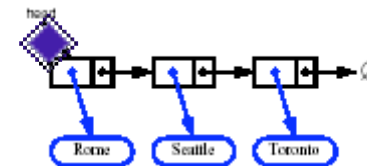
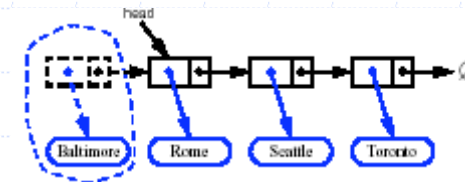
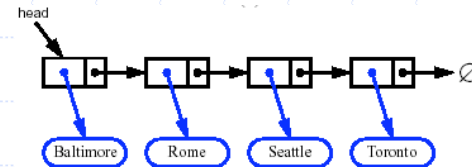
# Inserting at the Head

1. Allocate a new node
2. Insert new element
3. Have new node point to old head
4. Update head to point to new node



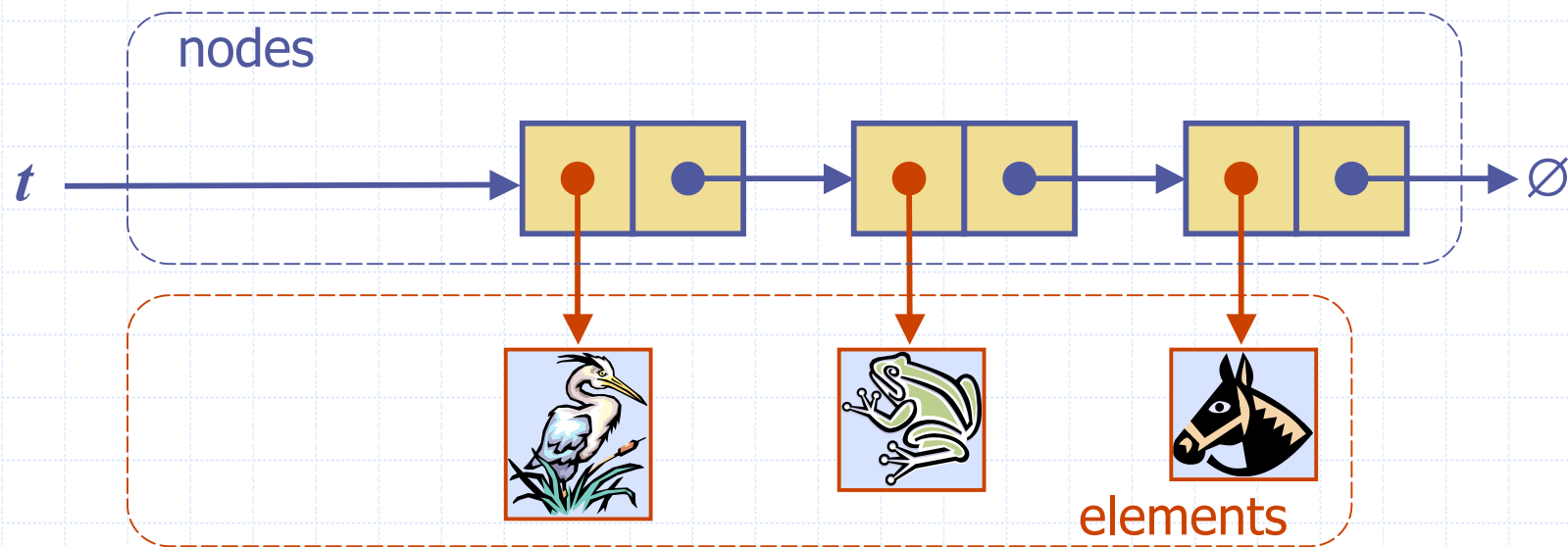
# Removing at the Head

1. Update head to point to next node in the list
2. Allow garbage collector to reclaim the former first node



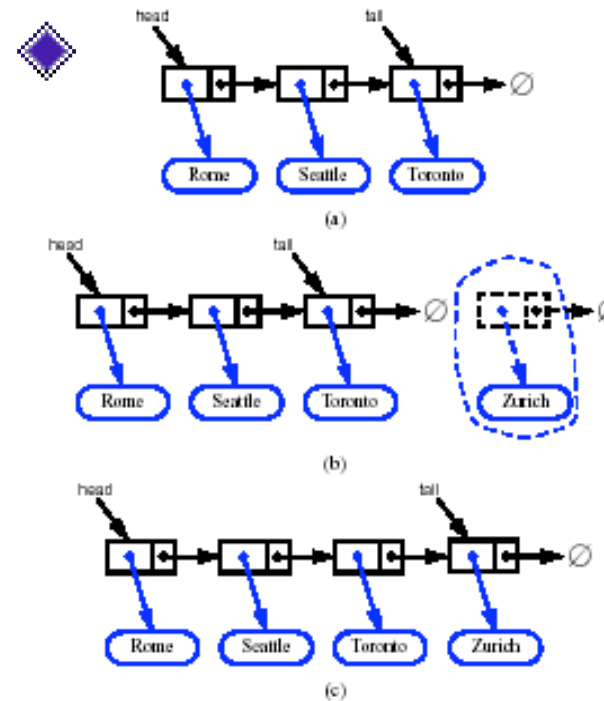
# Stack with a Singly Linked List

- ◆ We can implement a stack with a singly linked list
- ◆ The top element is stored at the first node of the list
- ◆ The space used is  $O(n)$  and each operation of the Stack ADT takes  $O(1)$  time



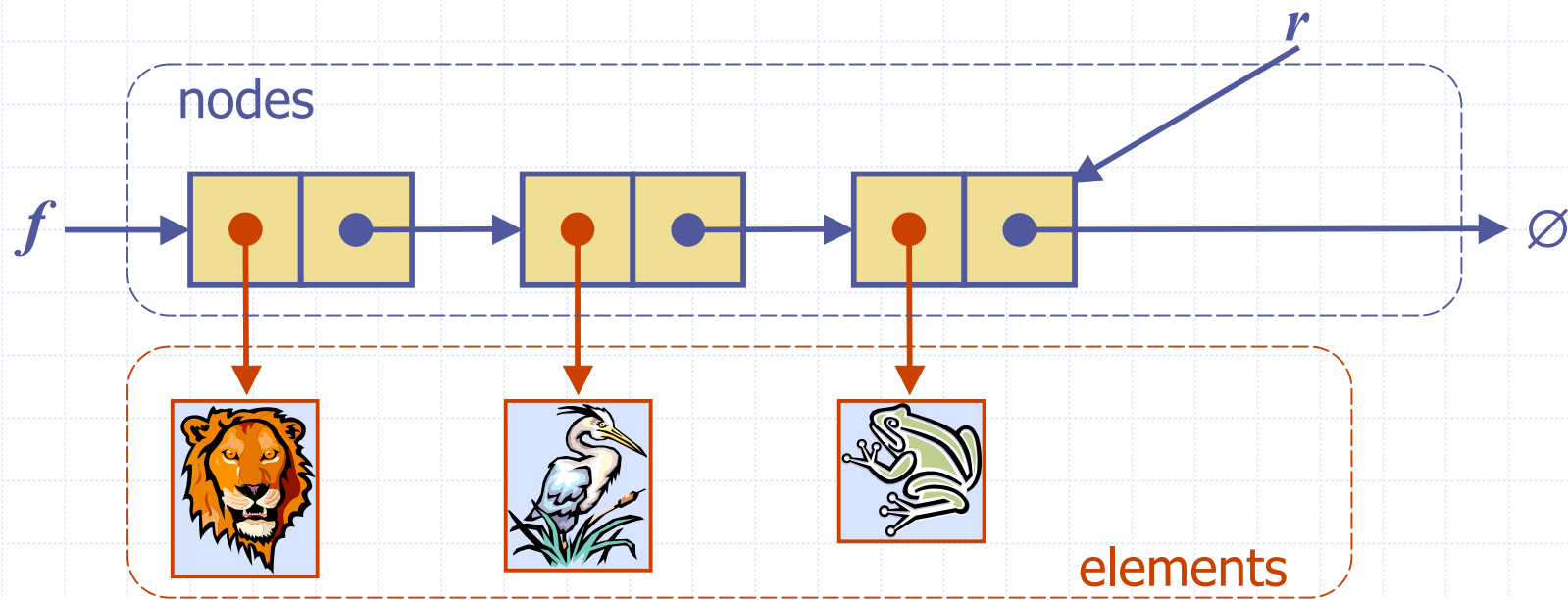
# Inserting at the Tail

1. Allocate a new node
2. Insert new element
3. Have new node point to null
4. Have old last node point to new node
5. Update tail to point to new node



# Queue with a Singly Linked List

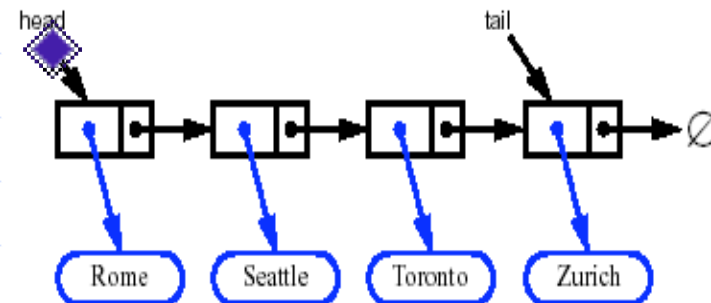
- ◆ We can implement a queue with a singly linked list
  - The front element is stored at the first node
  - The rear element is stored at the last node
- ◆ The space used is  $O(n)$  and each operation of the Queue ADT takes  $O(1)$  time





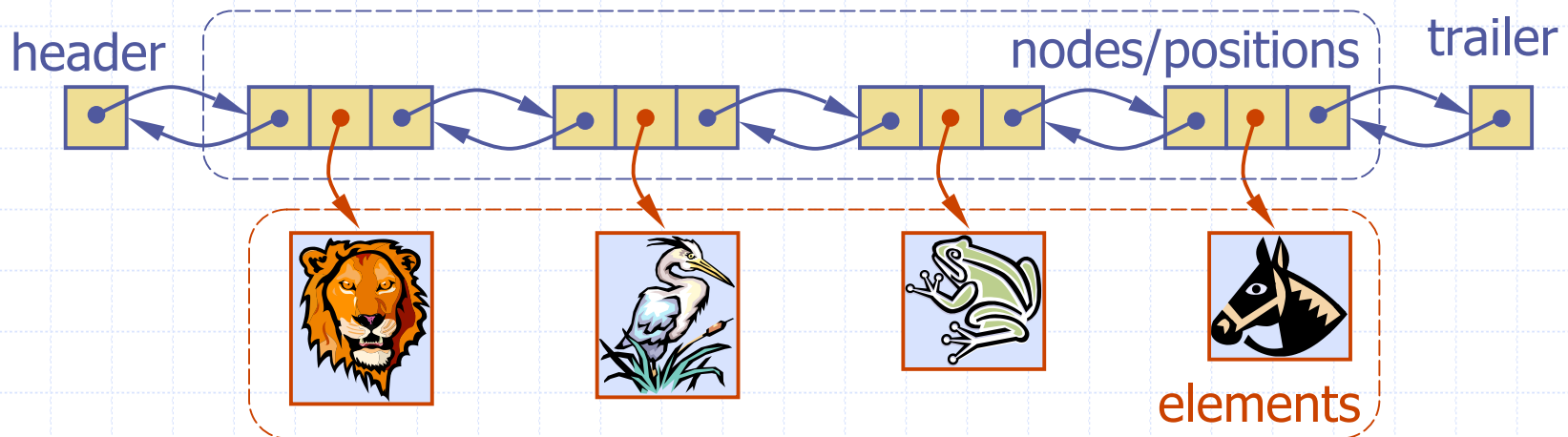
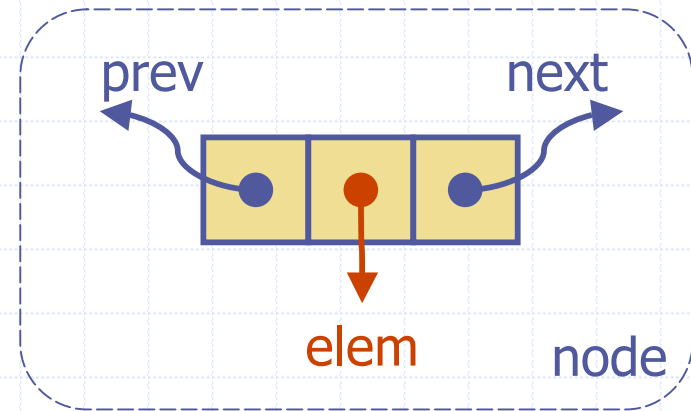
# Removing at the Tail

- ◆ Removing at the tail of a singly linked list is not efficient!
- ◆ There is no constant-time way to update the tail to point to the previous node



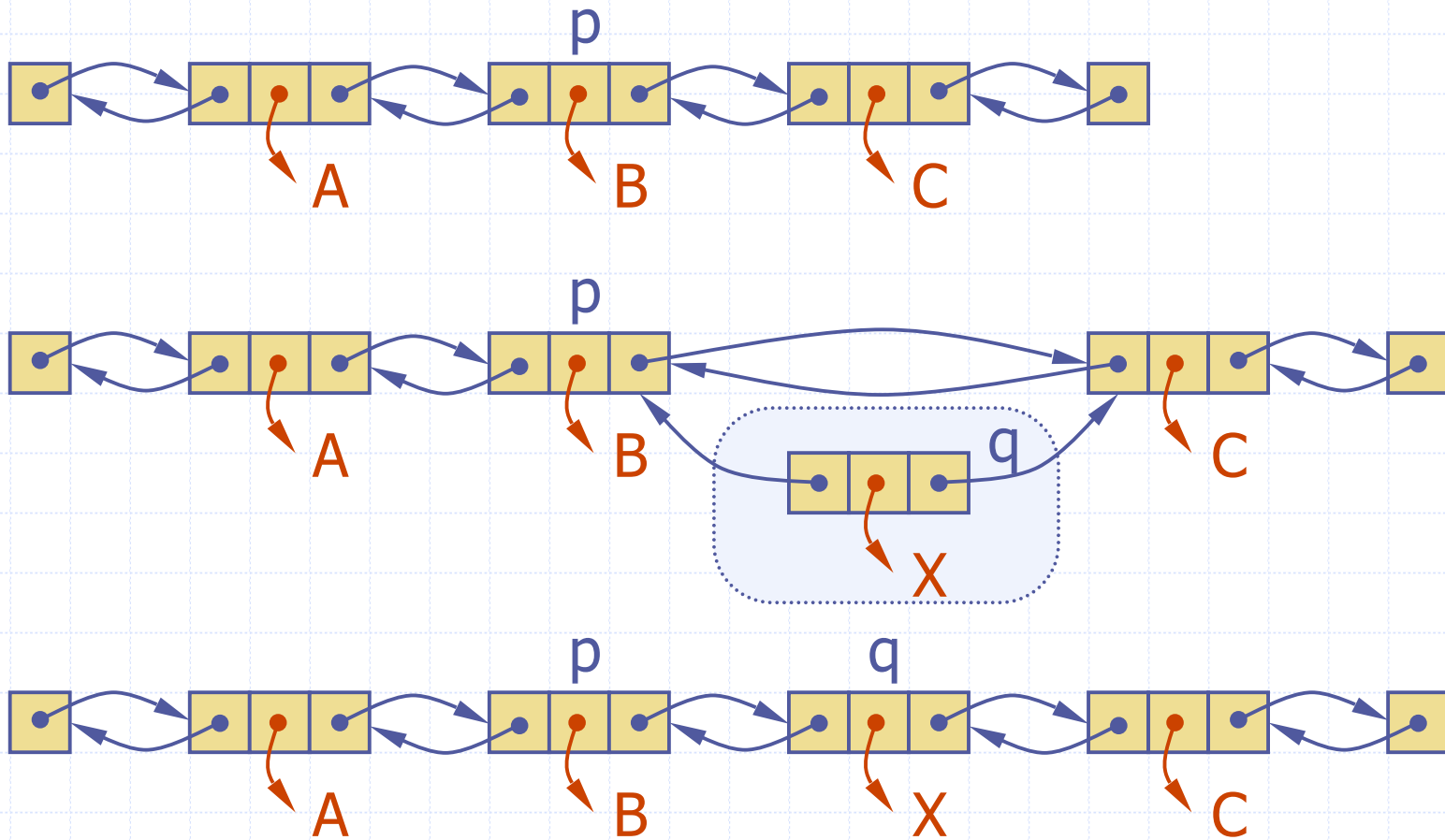
# Doubly Linked List

- ◆ A doubly linked list is a more complex type of list that allows insertion in the middle
- ◆ Nodes store:
  - element (sometimes combined into the node instead of a reference)
  - link to the previous node
  - link to the next node
- ◆ Special trailer and header nodes



# Insertion

- ◆ We visualize operation `insertAfter(p, X)`, which returns position `q`



# Deletion

- ◆ We visualize `remove(p)`, where `p = last()`

