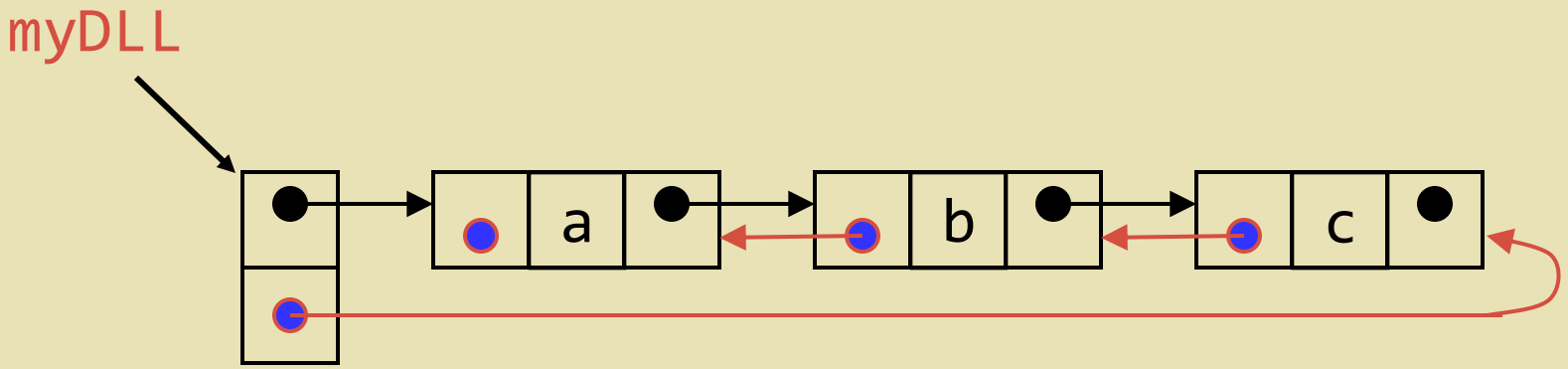
Doubly-linked Lists

Notes from Dr. Charlie Obimbo and revised by Yan Yan.

Doubly-linked lists

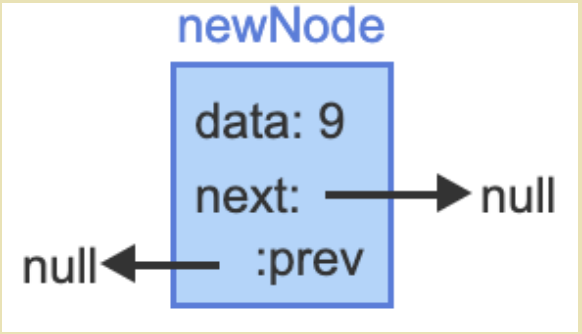
- Here is a doubly-linked list (DLL):

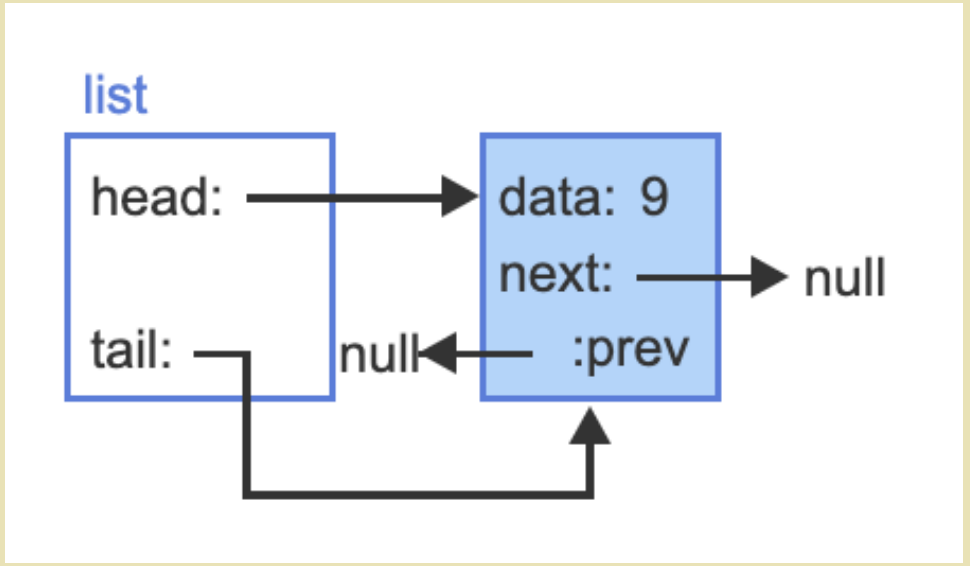


  - Each node contains a value, a link to its successor (if any), *and* a link to its predecessor (if any)
  - The head points to the first node in the list *and* to the last node in the list (or contains null links if the list is empty). The two pointers are also called head and tail.
  - DLL structure -- *two* pointers for each node

Doubly-linked lists

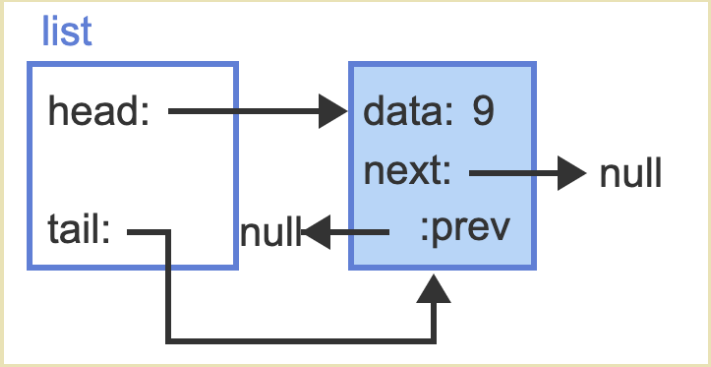
- **Append**: insert after the list's tail node
  - Empty list -- point the list's head and tail pointers to the new node.

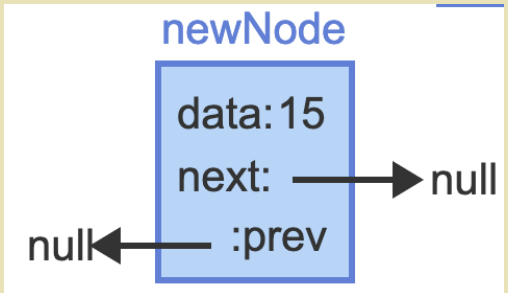


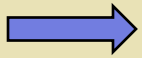


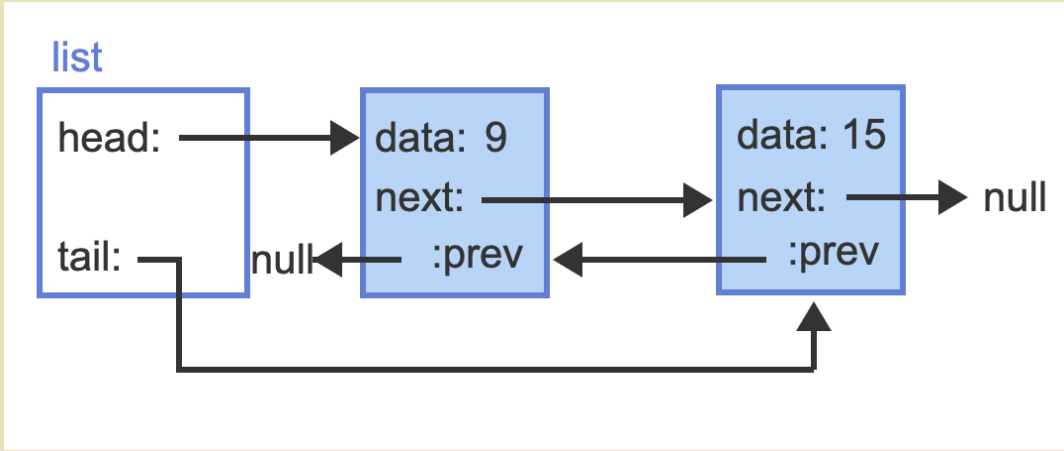
Doubly-linked lists

- **Append**: insert after the list's tail node
  - Non-empty list -- point the tail node's next pointer to the new node, point the new node's previous pointer to the list's tail node, and point the list's tail pointer to the new node.







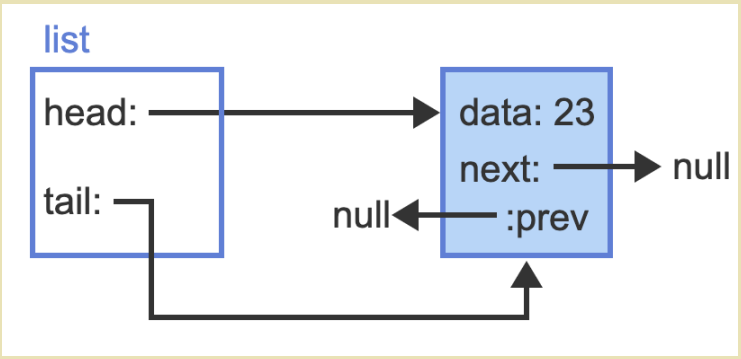


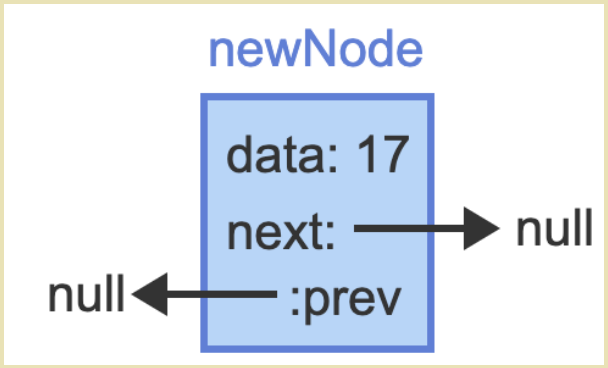
Doubly-linked lists

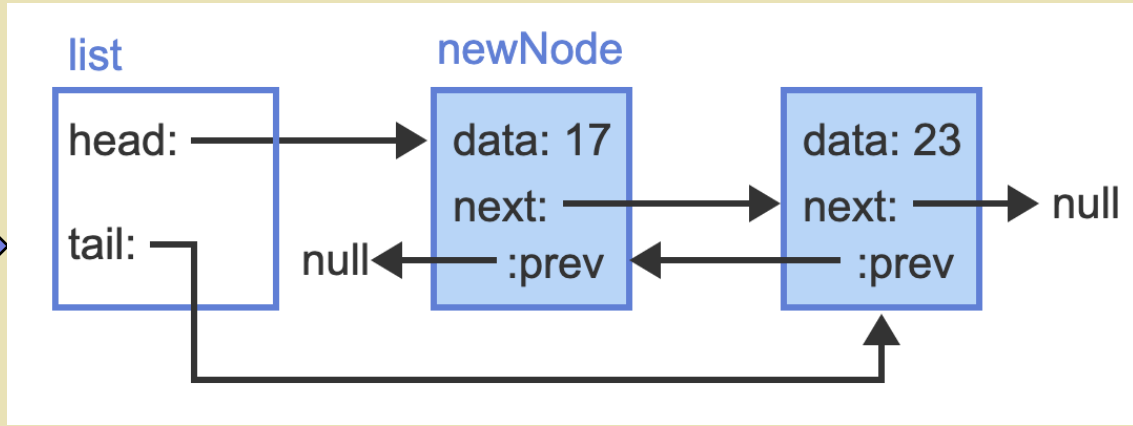
- **Prepend**: insert before the list's head node and points the head pointer to the new node.
  - Empty list -- point the list's head and tail pointers to the new node..
  - Non-empty list -- point the new node's next pointer to the list's head node, point the list head node's previous pointer to the new node, and point the list's head pointer to the new node

Doubly-linked lists

- **Prepend**: insert before the list's head node and points the head pointer to the new node.
  - Non-empty list example

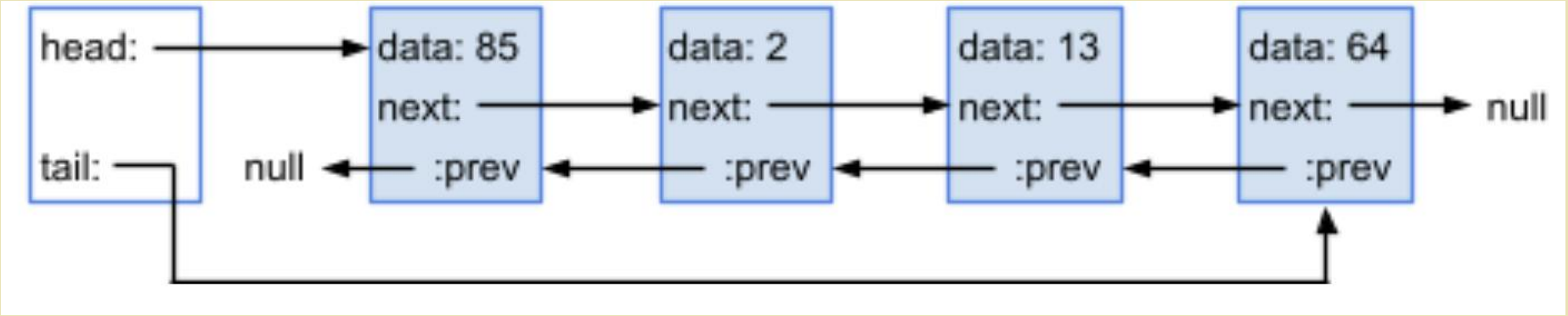






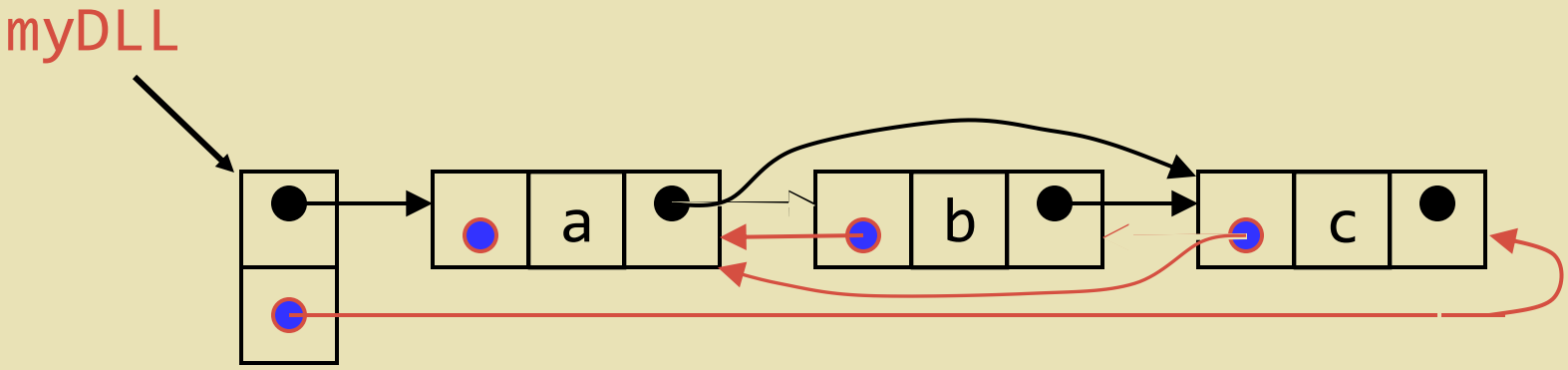
Doubly-linked lists

- **Reverse Traversal**:
  - DLL supports a reverse traversal. It visits all nodes starting with the list's tail node and ending after visiting the list's head node. node.



Deleting a node from a DLL

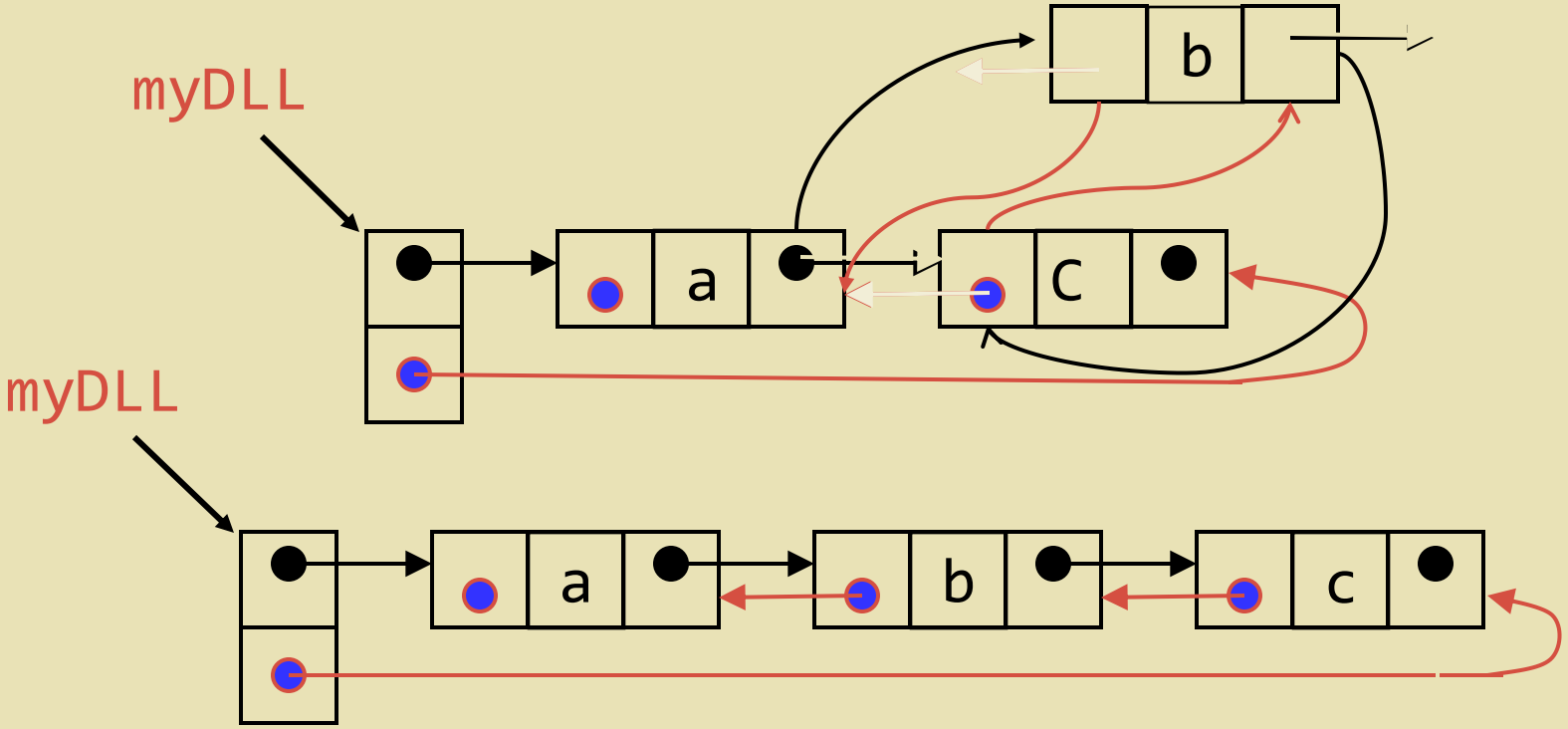
- Node deletion from a DLL involves changing *two* links
- In this example,we will delete node b



- We don’t have to do anything about the links in node b
- Garbage collection will take care of deleted nodes
- Deletion of the first node or the last node is a special case

Inserting a node from a DLL

- Node insertion from a DLL involves changing *four* links
- In this example,we will insert node b



- Insertion after the last is Append, and before the first is Prepend.

DLLs compared to SLLs

- Advantages:
  - Can be traversed in either direction (may be essential for some programs)
  - Some operations, such as *deletion* and *inserting* before a node, become easier
- Disadvantages:
  - Requires more space
  - List manipulations are slower (because more links must be changed)
  - Greater chance of having bugs (because more links must be manipulated)

Other operations on linked lists

- Most “algorithms” on linked lists—such as insertion, deletion, and searching—are pretty obvious; you just need to be careful
- Sorting a linked list is just messy, since you can’t directly access the n^{th} element—you have to count your way through a lot of other elements

Resources on CourseLink

- C Programming
- Big-O

That’s about this lecture!