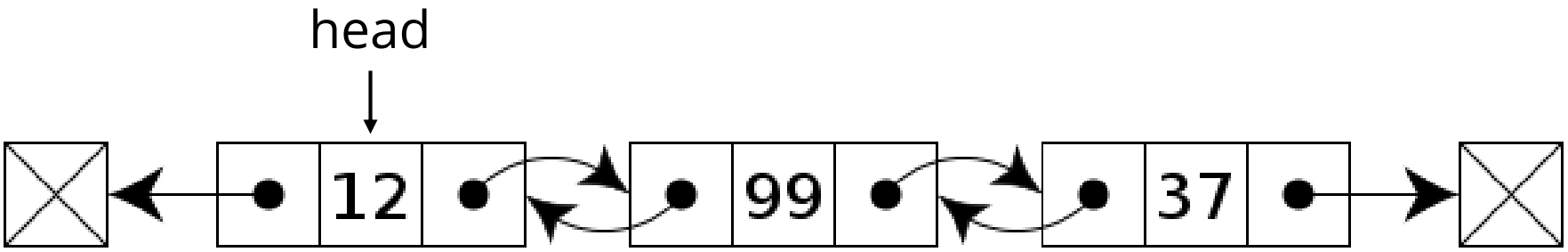
Doubly Linked List

“(Almost) Double the effort, half the trouble”

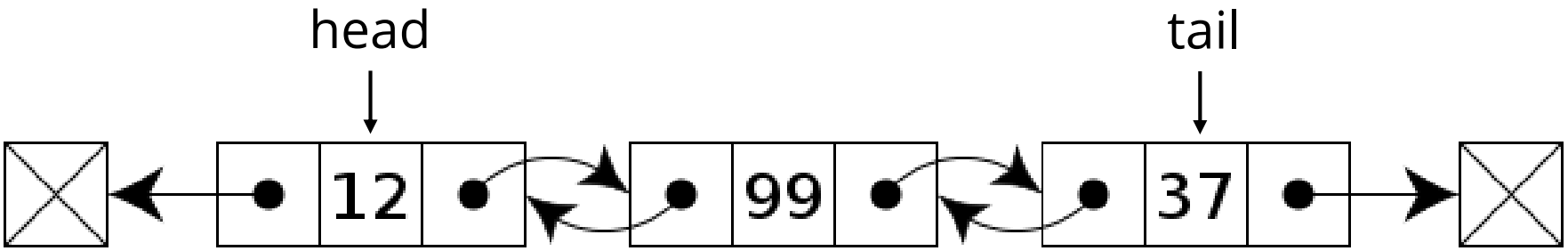
Prerequisite: Single Linked List

Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

Doubly Linked List

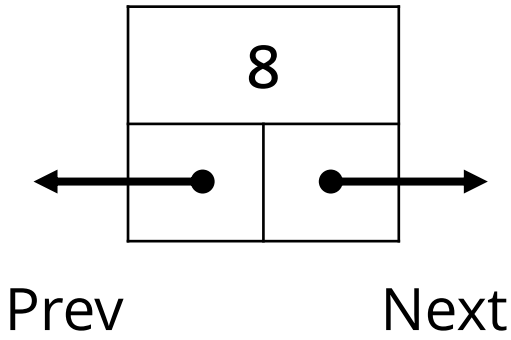


Doubly Linked List



Doubly Linked List

A Node



Traits

Pros

1. Can traverse both in forward and backward direction
2. Delete Operation in more efficient

Cons

1. For every node, two extra information are required
2. Have to maintain previous pointer and tail on top of next

Operations

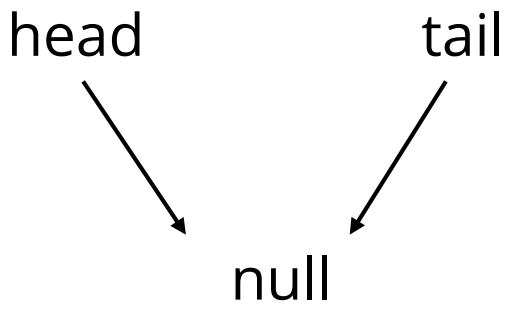
1. Insertion
    1. At front (push_front)
    2. At end (push_back)
    3. After a given node (pointer)
    4. Before a given node (pointer)

2. Deletion
    1. At front (pop_front)
    2. At end (pop_back)
    3. By value
    4. By node (pointer)

3. Search

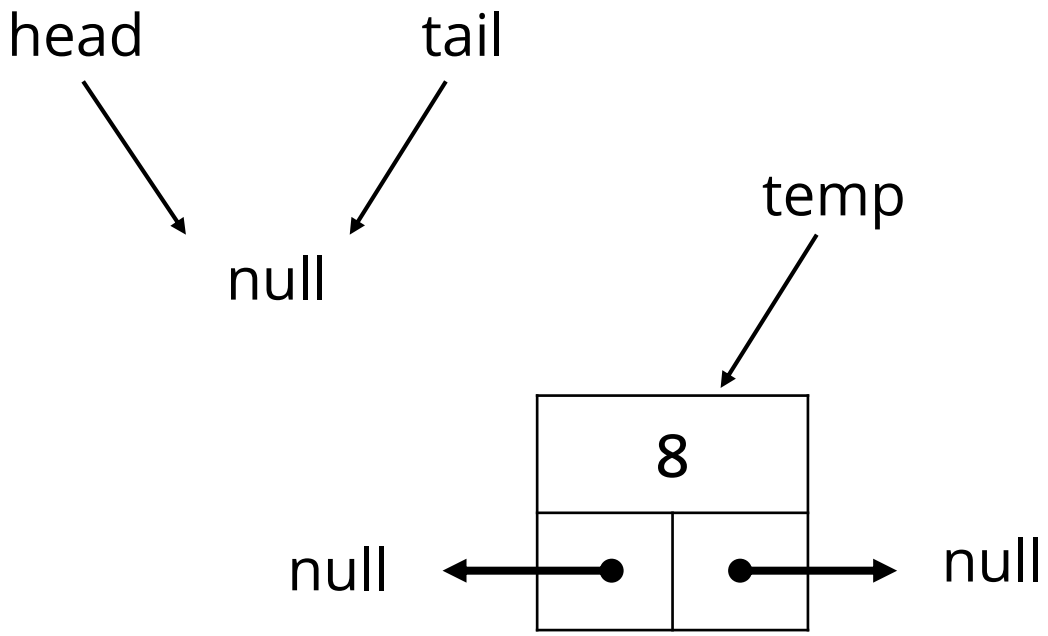
Insertion

At front



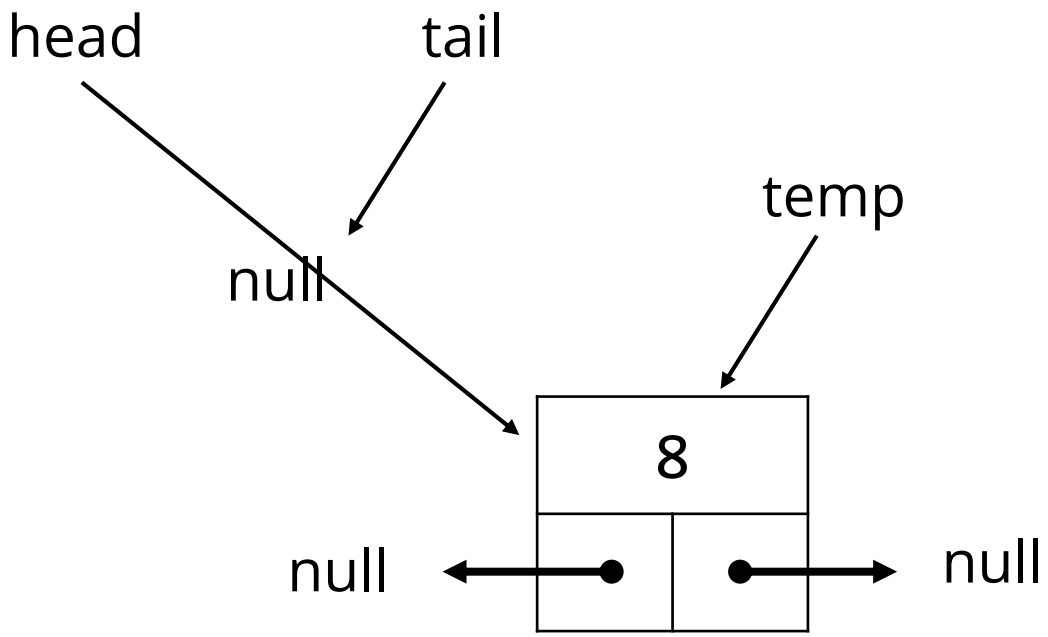
Insertion

At front



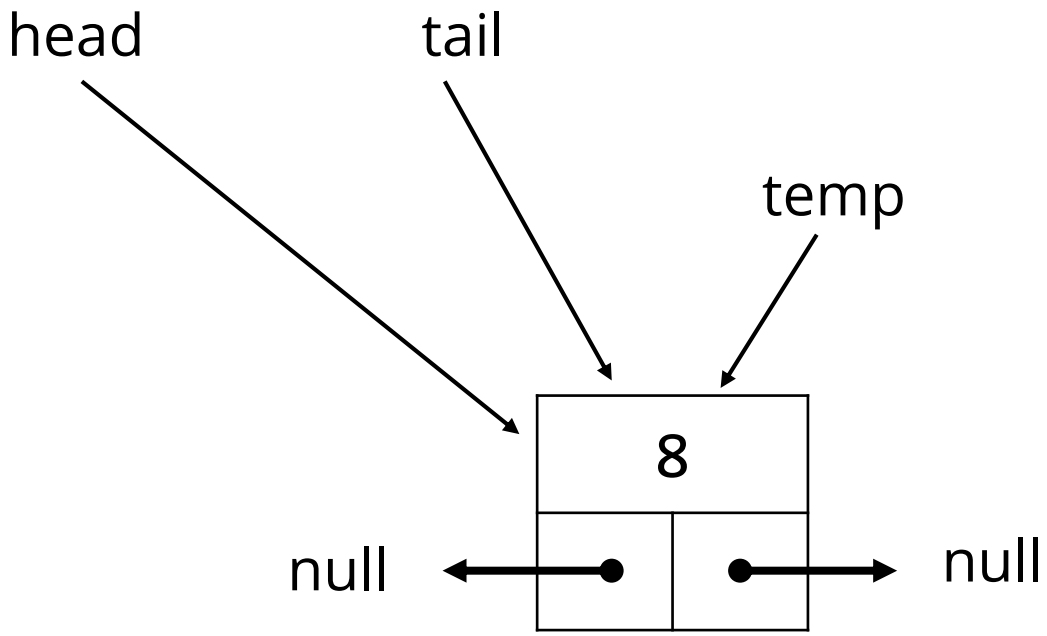
Insertion

At front



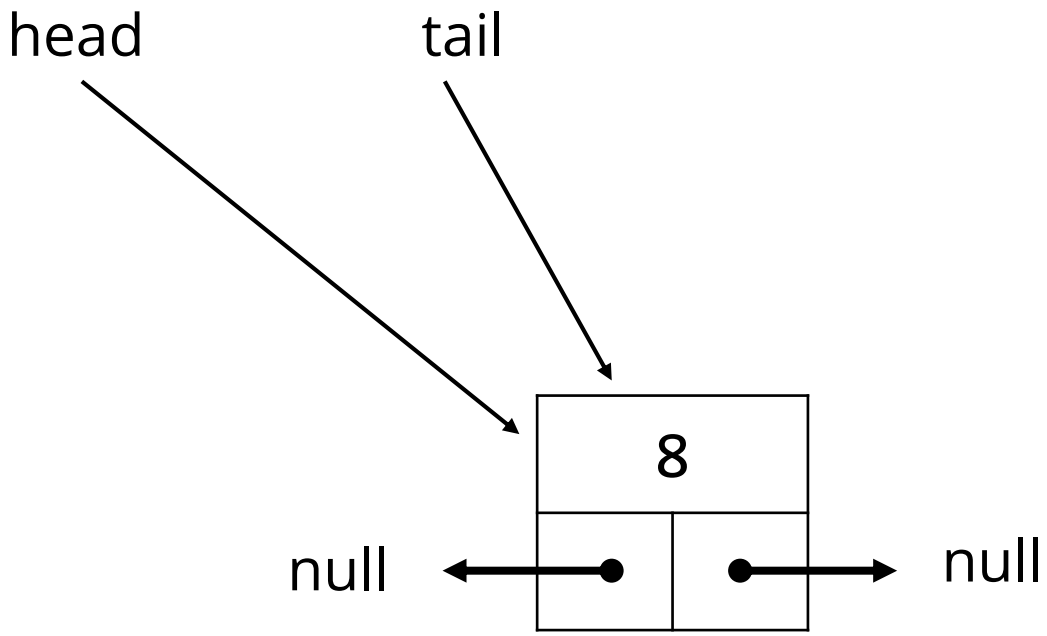
Insertion

At front



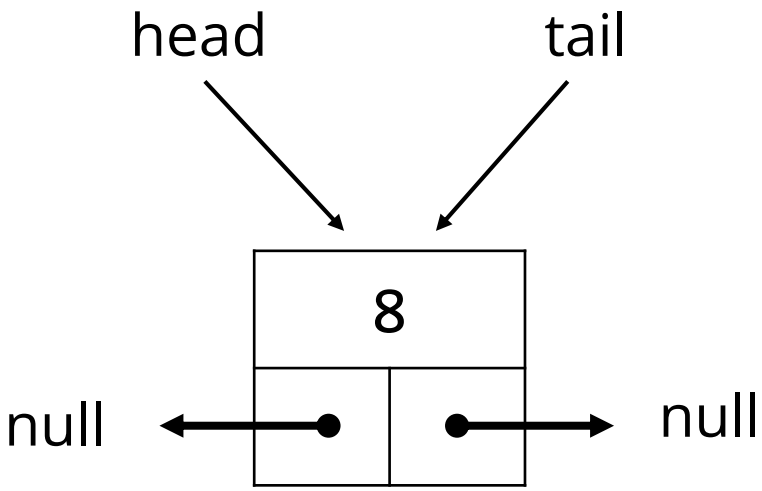
Insertion

At front



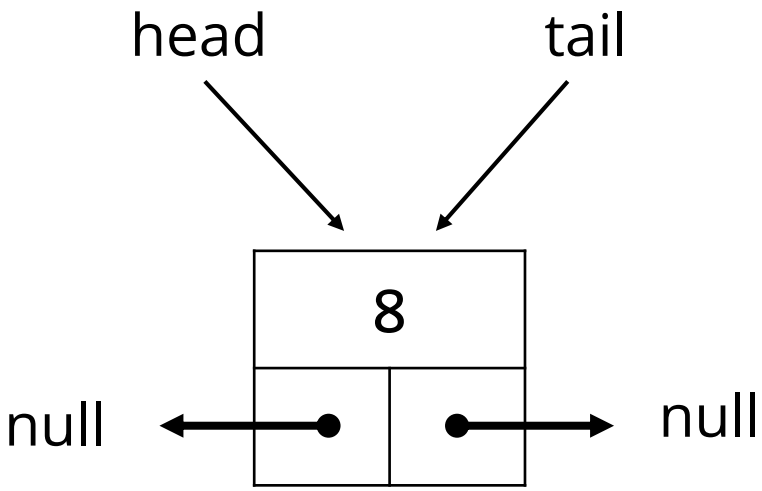
Insertion

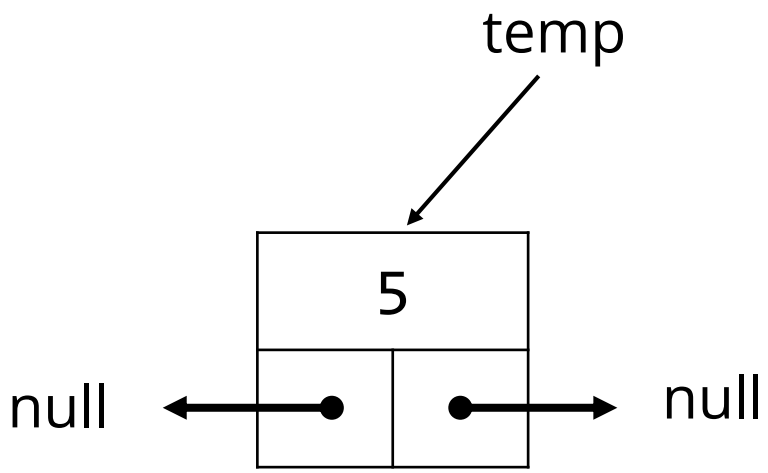
At front



Insertion

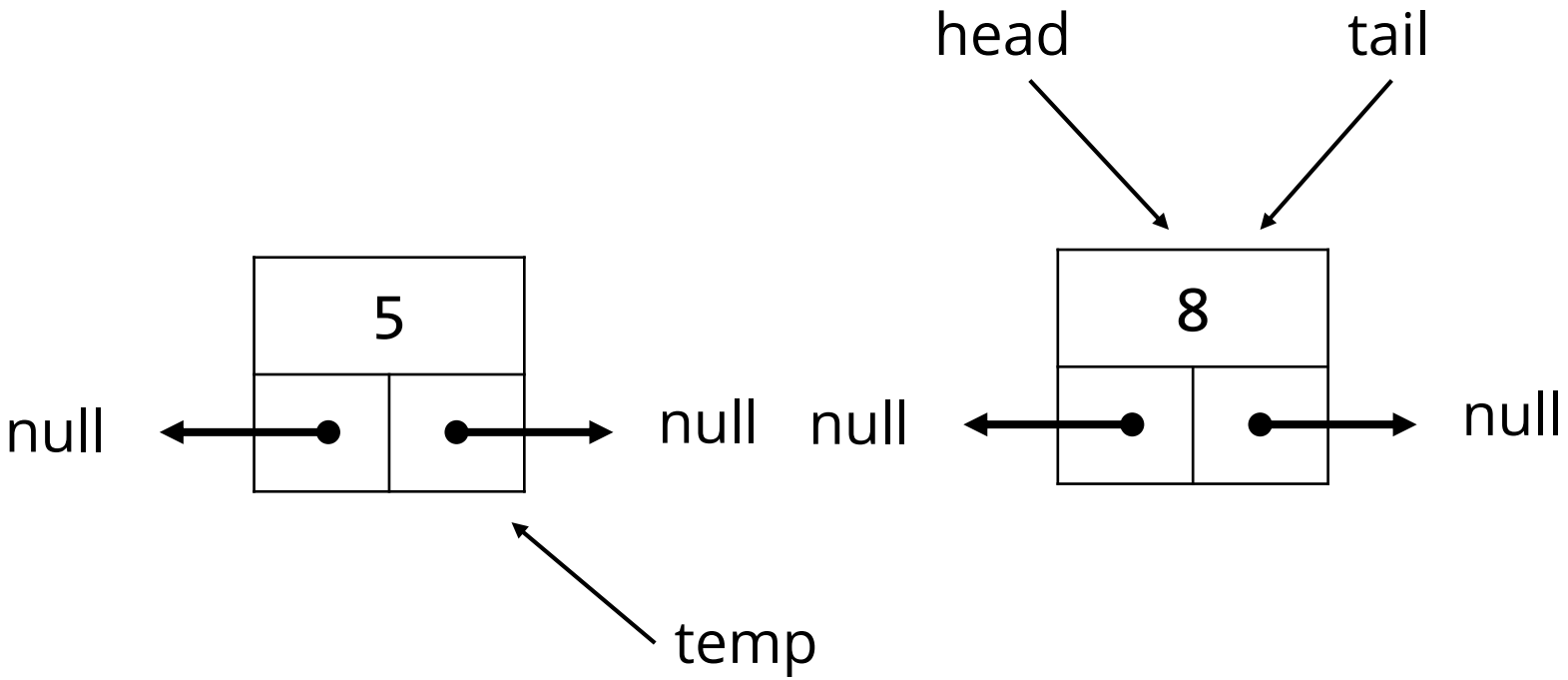
At front





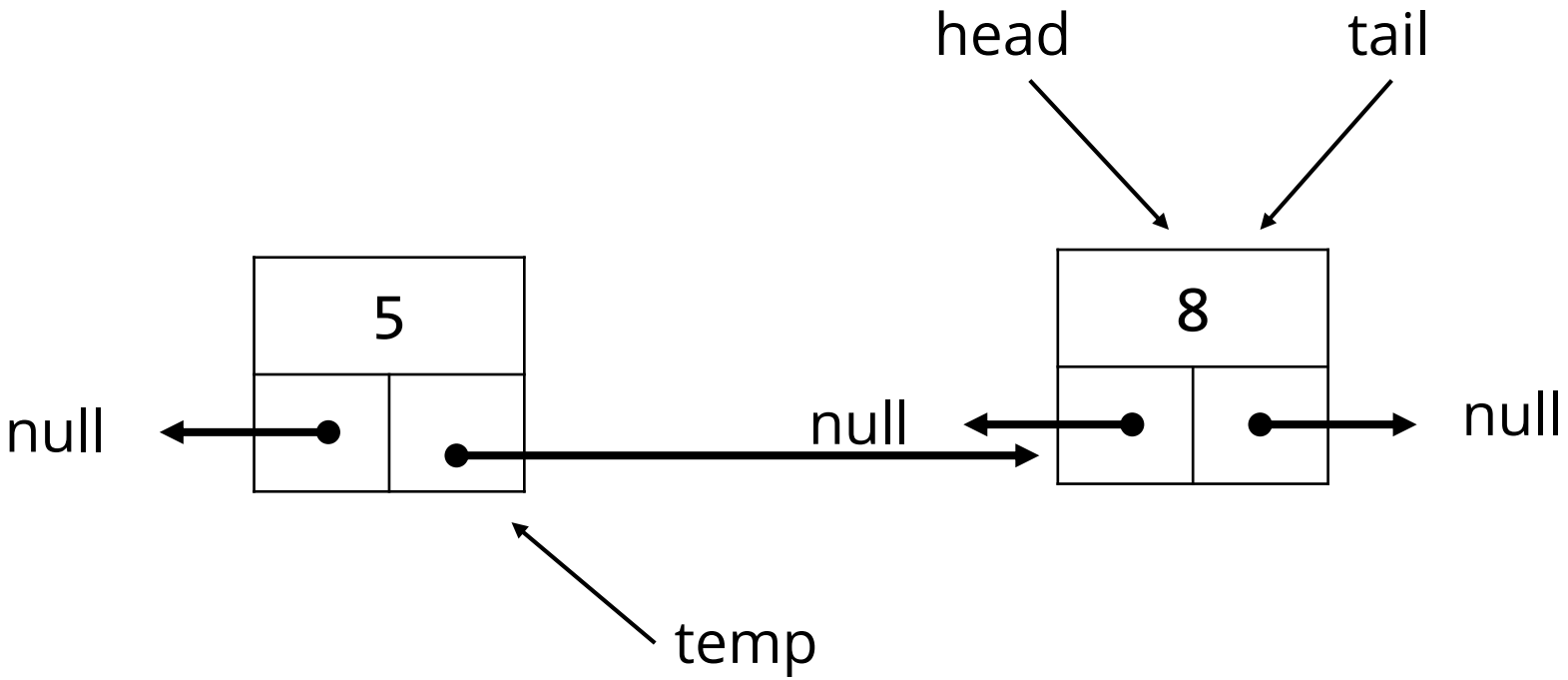
Insertion

At front



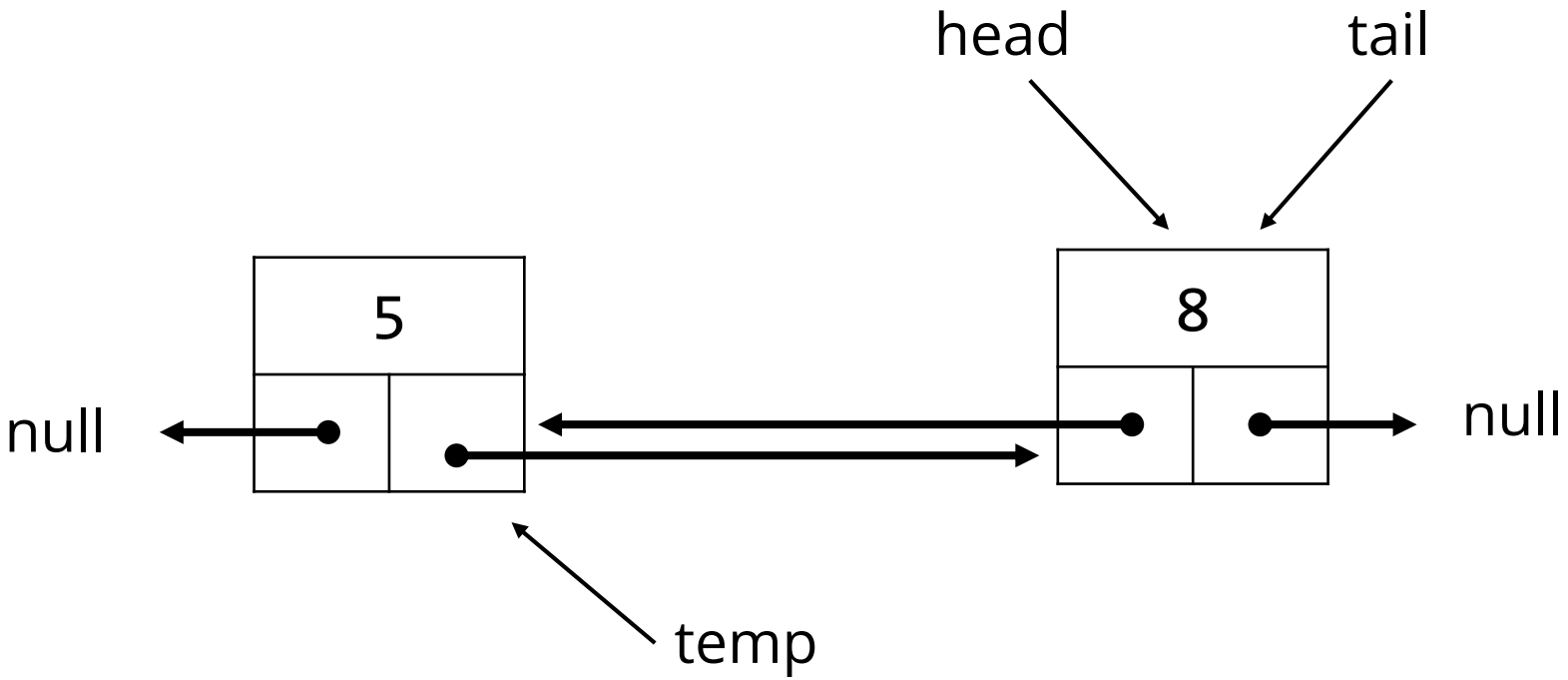
Insertion

At front



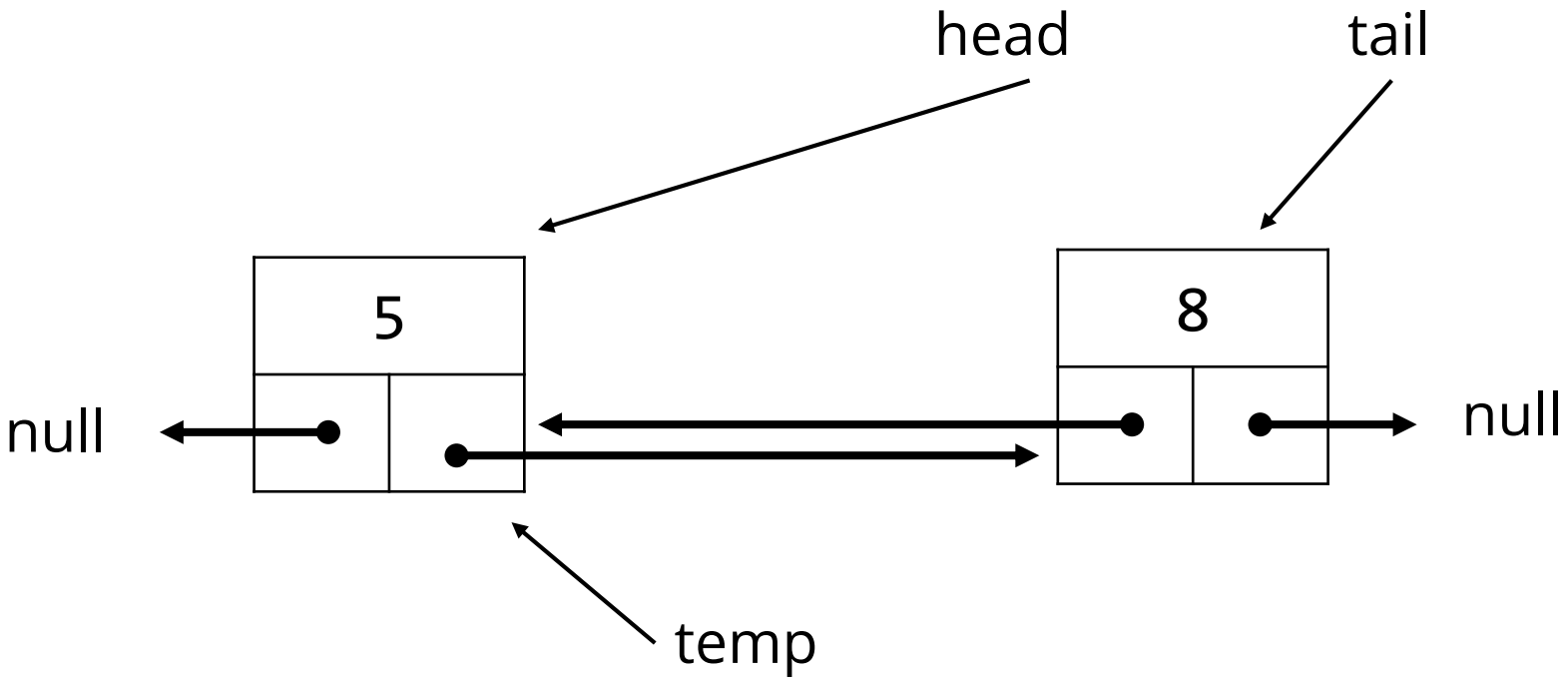
Insertion

At front



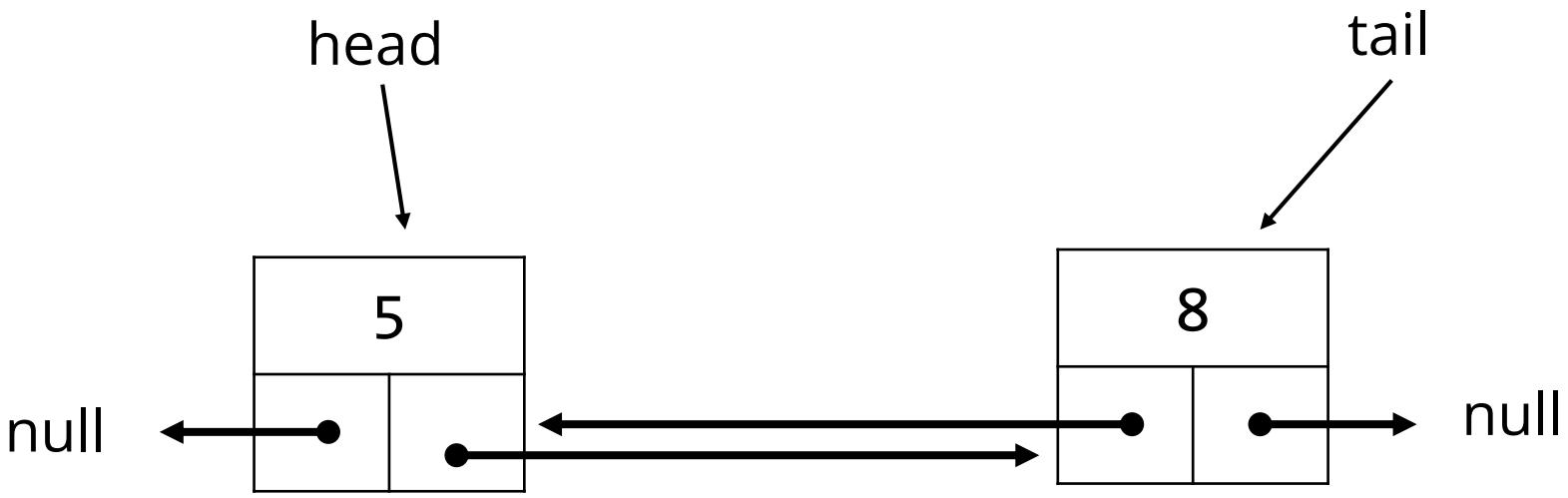
Insertion

At front



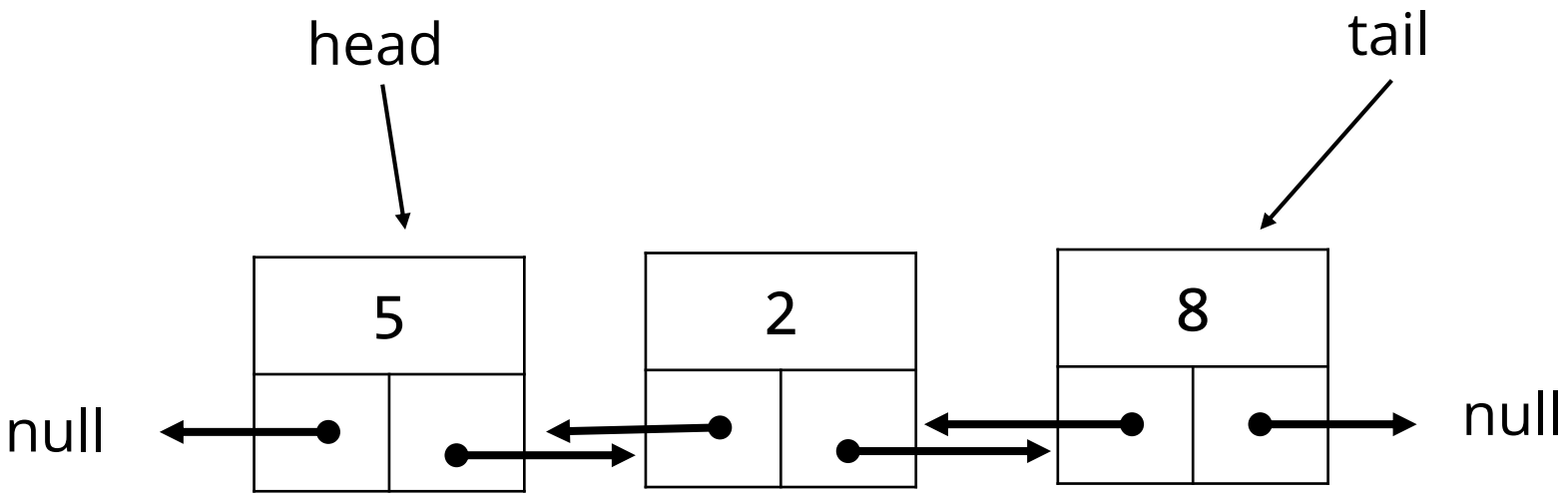
Insertion

At front



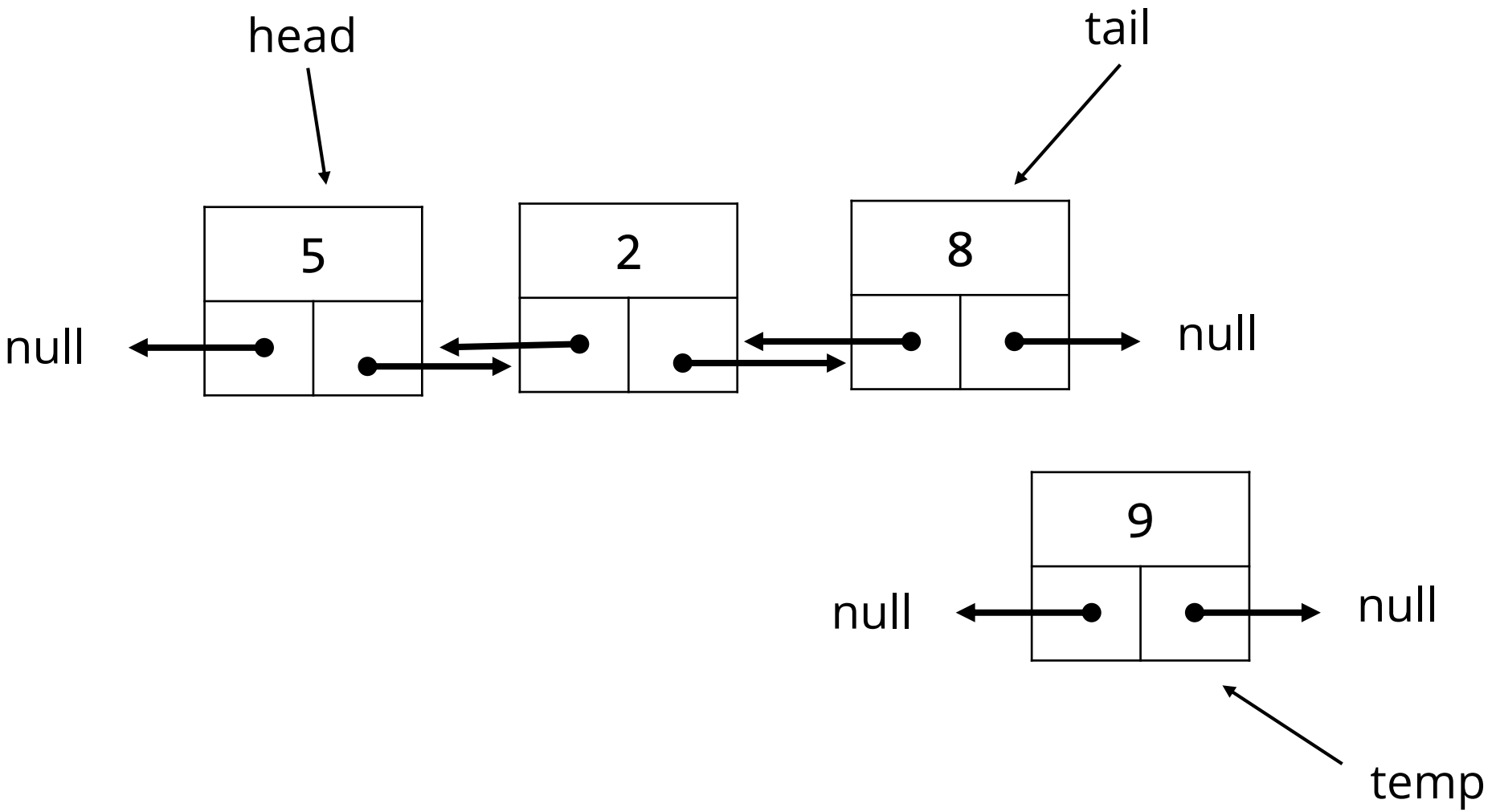
Insertion

At end



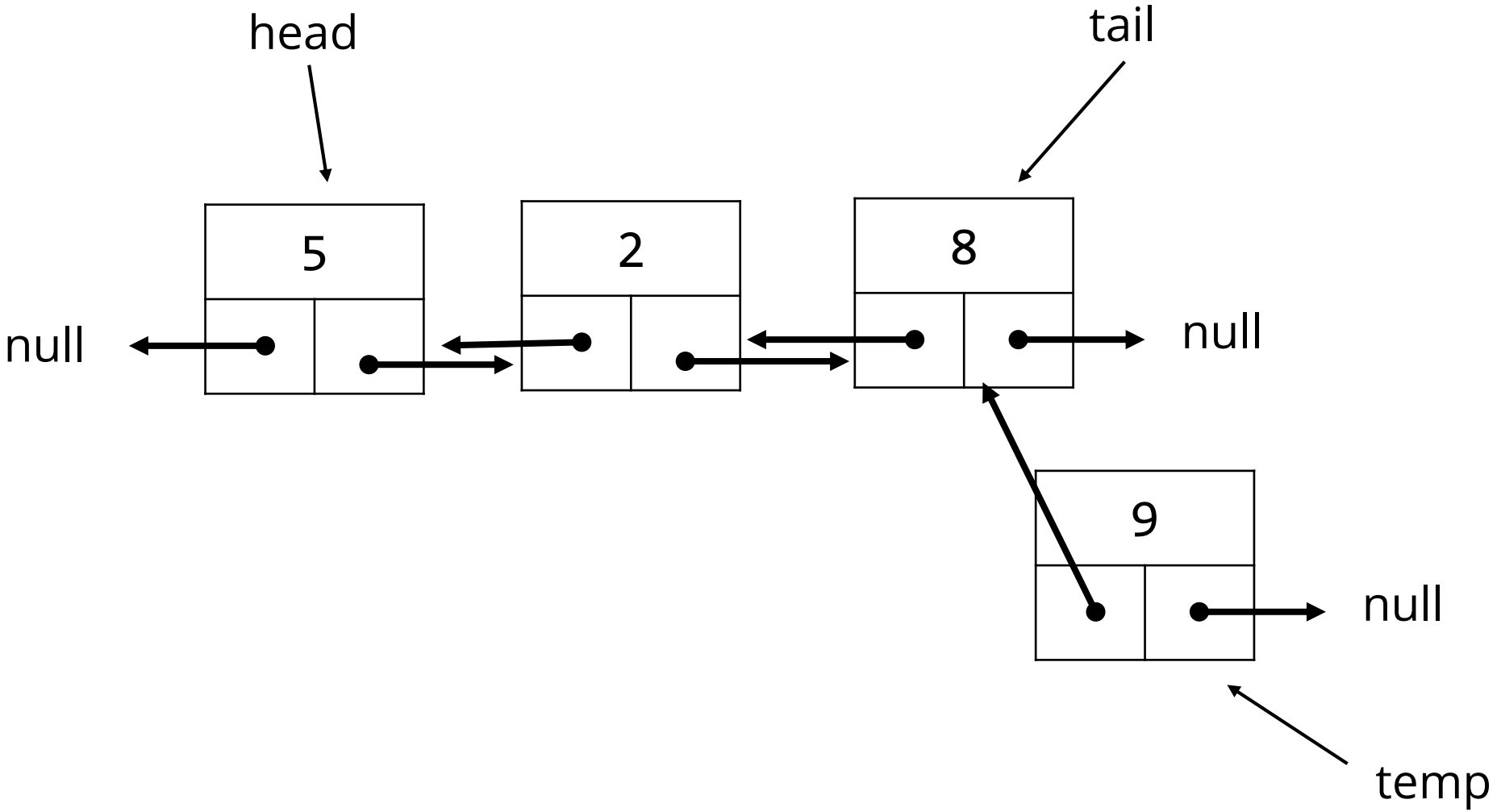
Insertion

At end



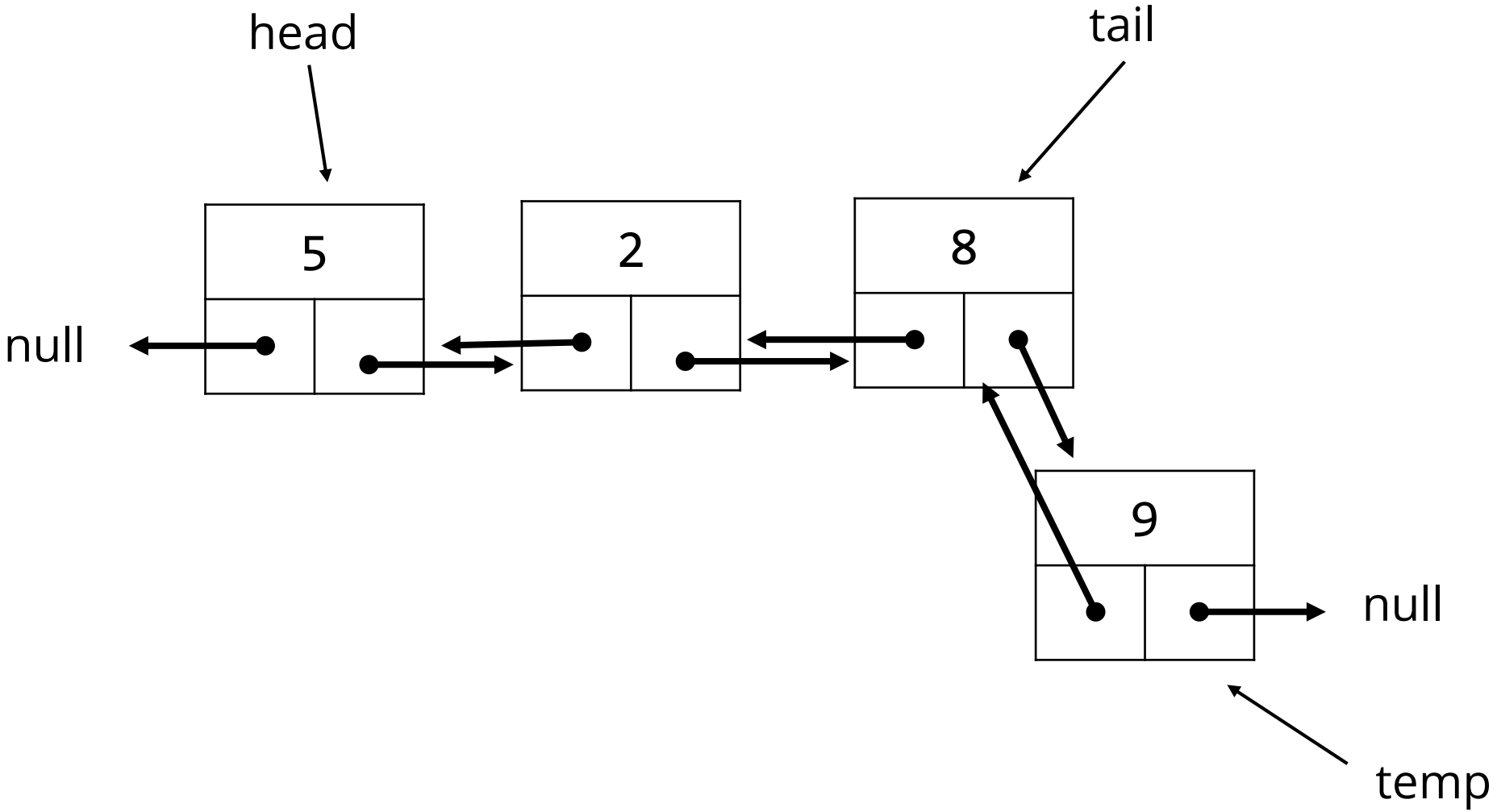
Insertion

At end



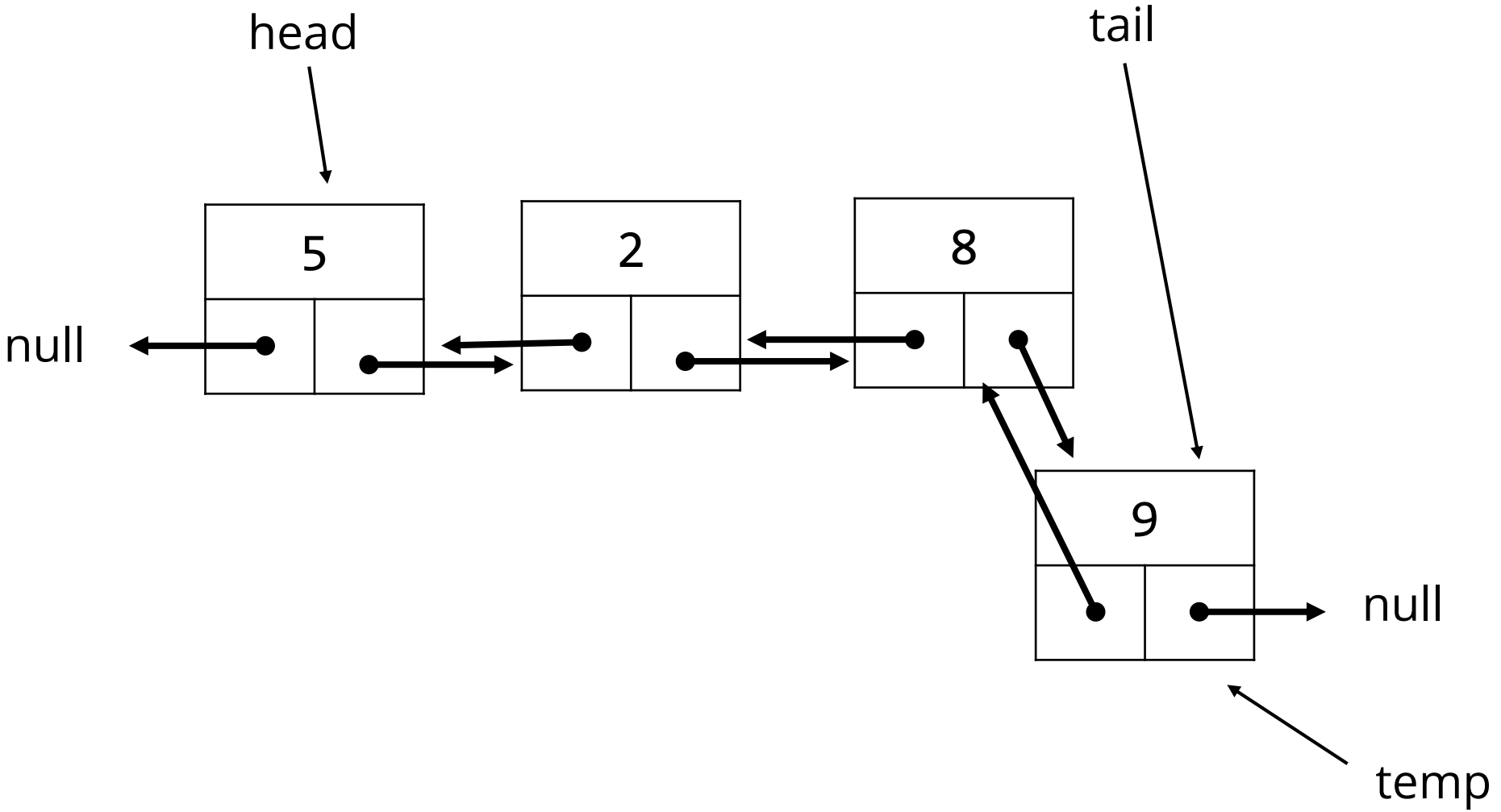
Insertion

At end



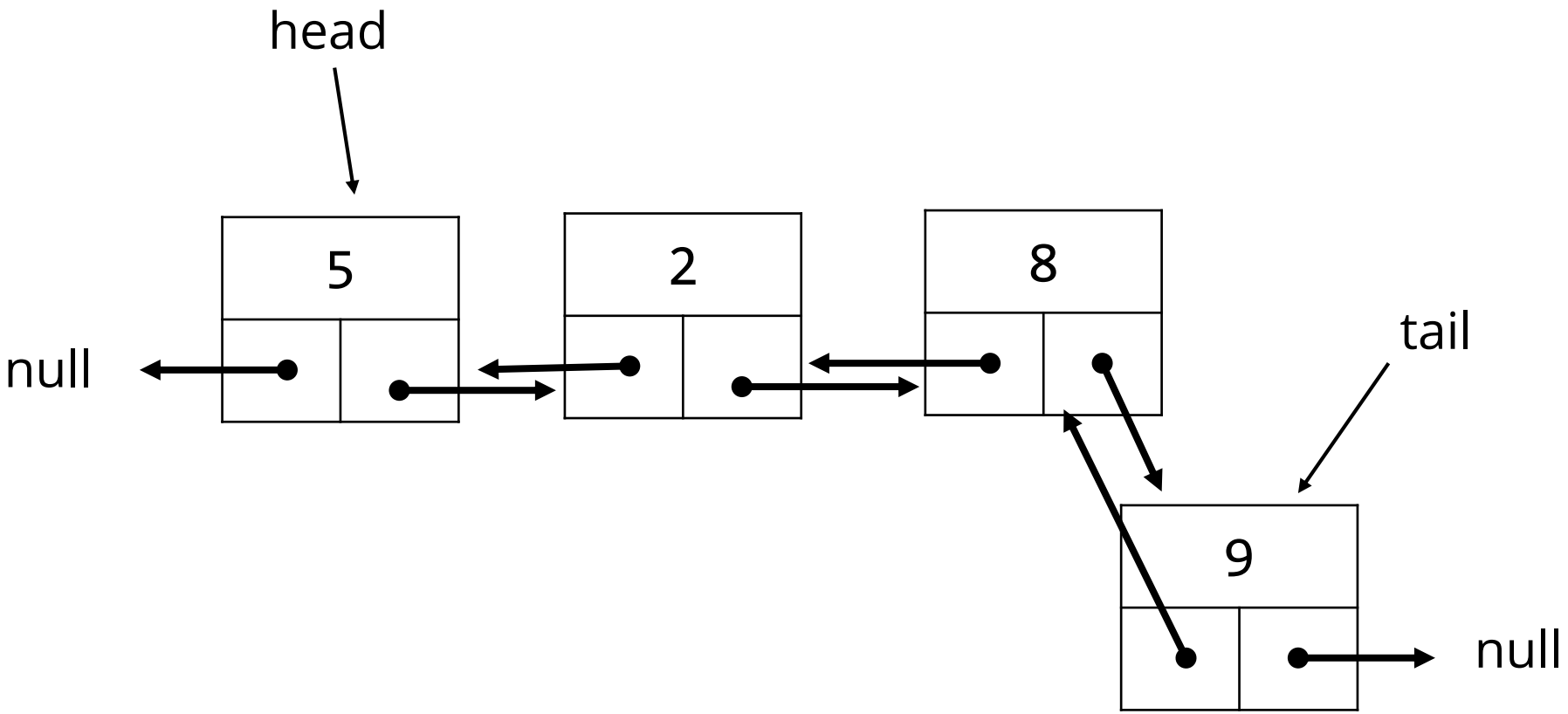
Insertion

At end



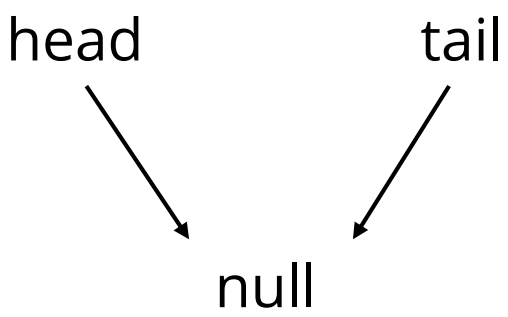
Insertion

At end



What if the list is initially empty?

How to add at end?



Exercise

1. Implement *Search*

Exercise

1. Implement *Search*
2. Implement *Add after a value*

Reference

1. https://www.geeksforgeeks.org/doubly-linked-list/

2. https://en.wikipedia.org/wiki/Doubly_linked_list

3. https://www.geeksforgeeks.org/delete-a-node-in-a-doubly-linked-list/