Single Linked List Operations

“Sword-fighting using pointers”

Prerequisite: Pointer, Structure

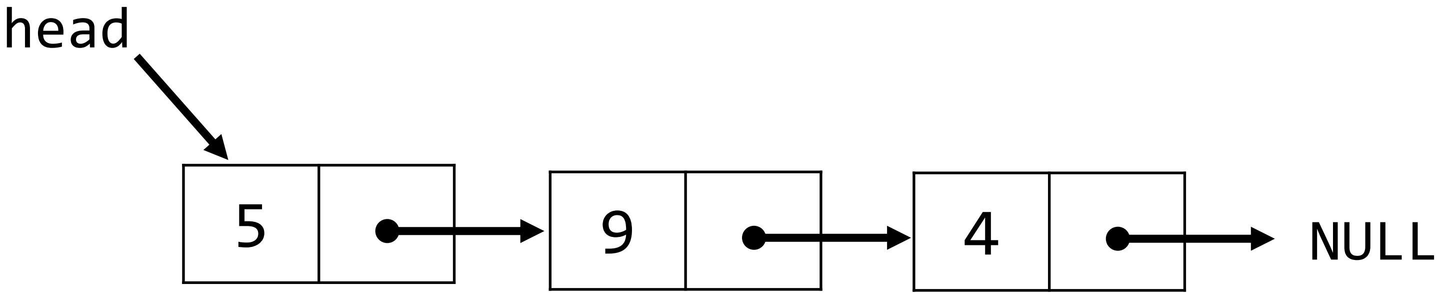
Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

Standard Operations of Linked List

1. Insertion − Adds an element at the beginning of the list.
2. Deletion − Deletes an element at the beginning of the list.
3. Display − Displays the complete list.
4. Search − Searches an element using the **given key**.
5. Delete − Deletes an element using the **given key**.
6. Clear All – Dispose all elements.
7. Other operations: size(), add_after(value1, value2)

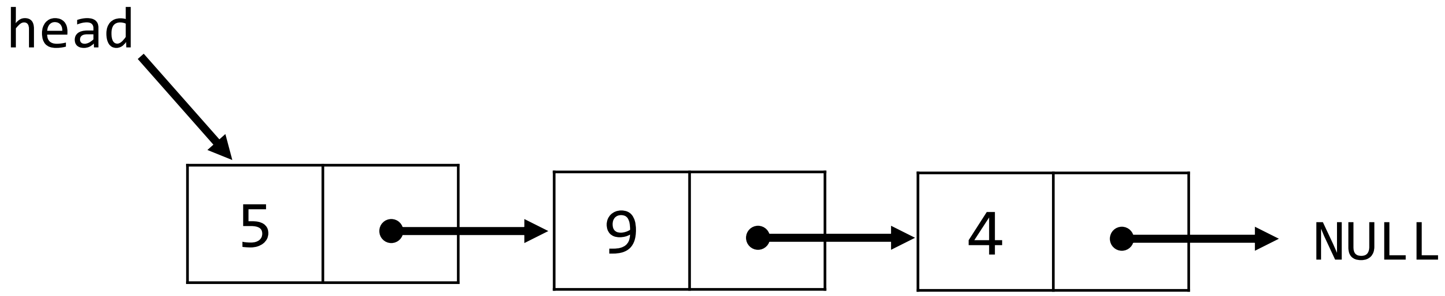
Source: https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm

Single Linked List



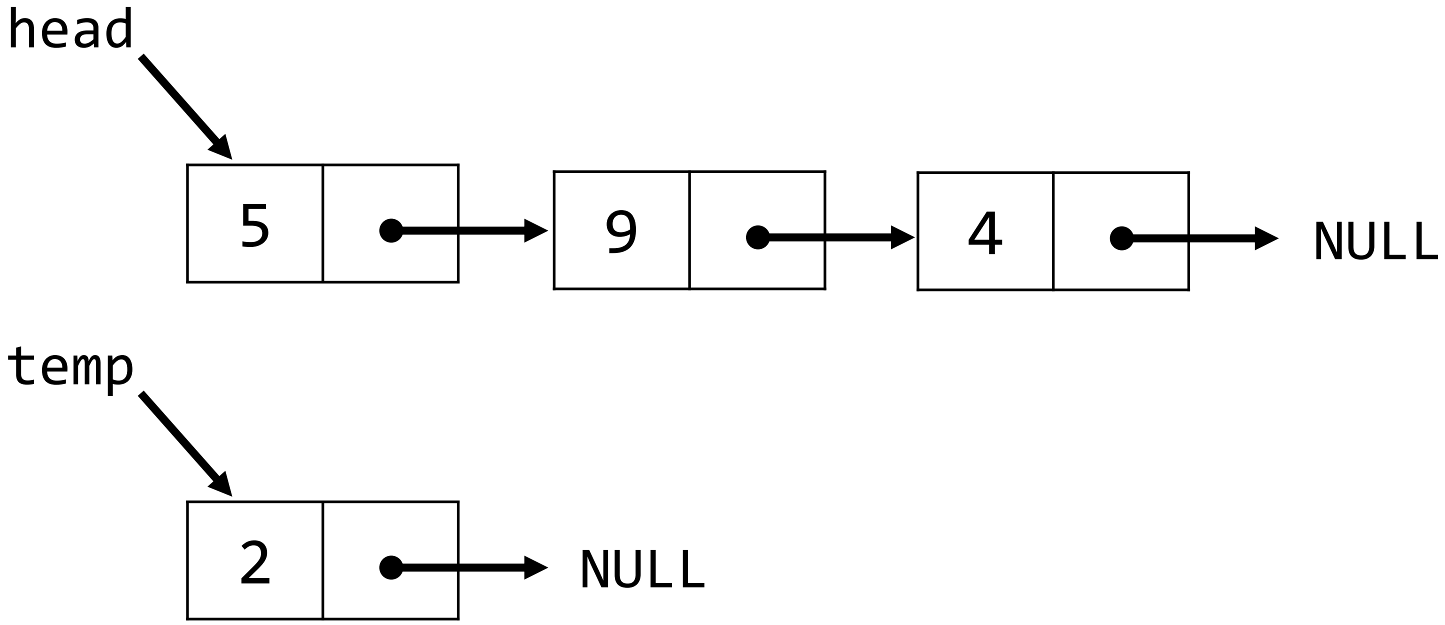
Single Linked List

push_front() operation



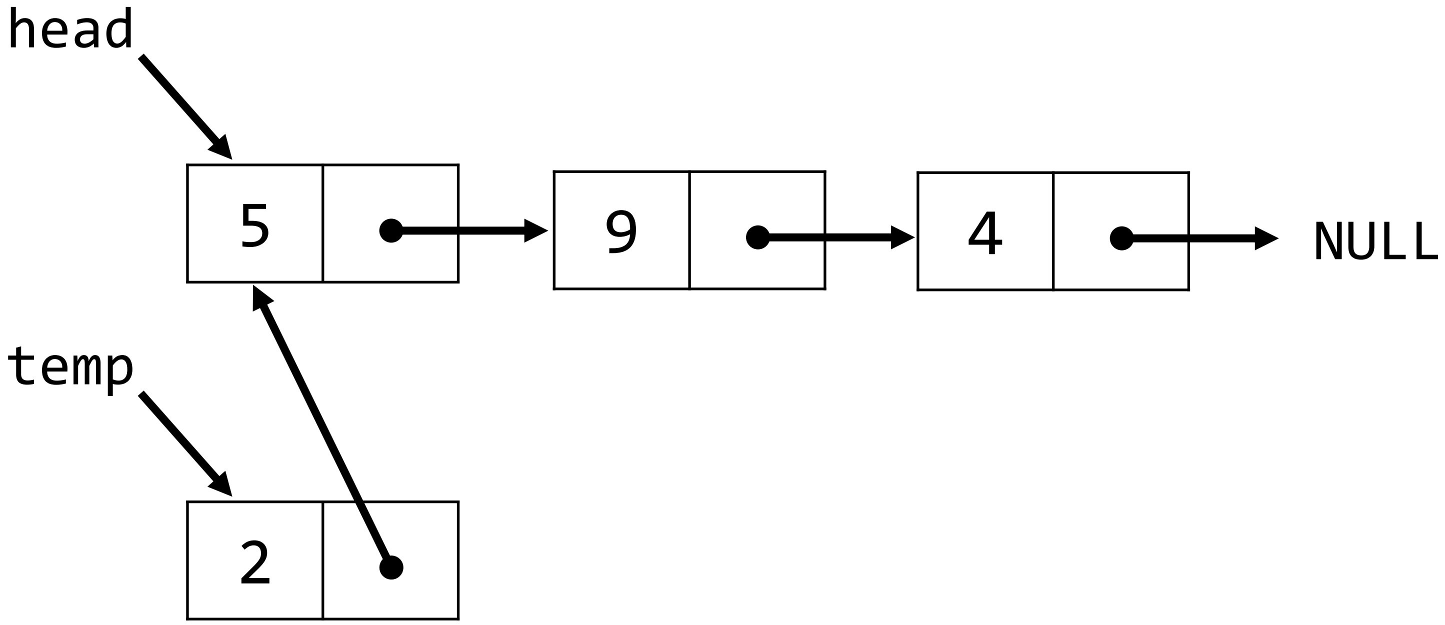
Single Linked List

push_front() operation



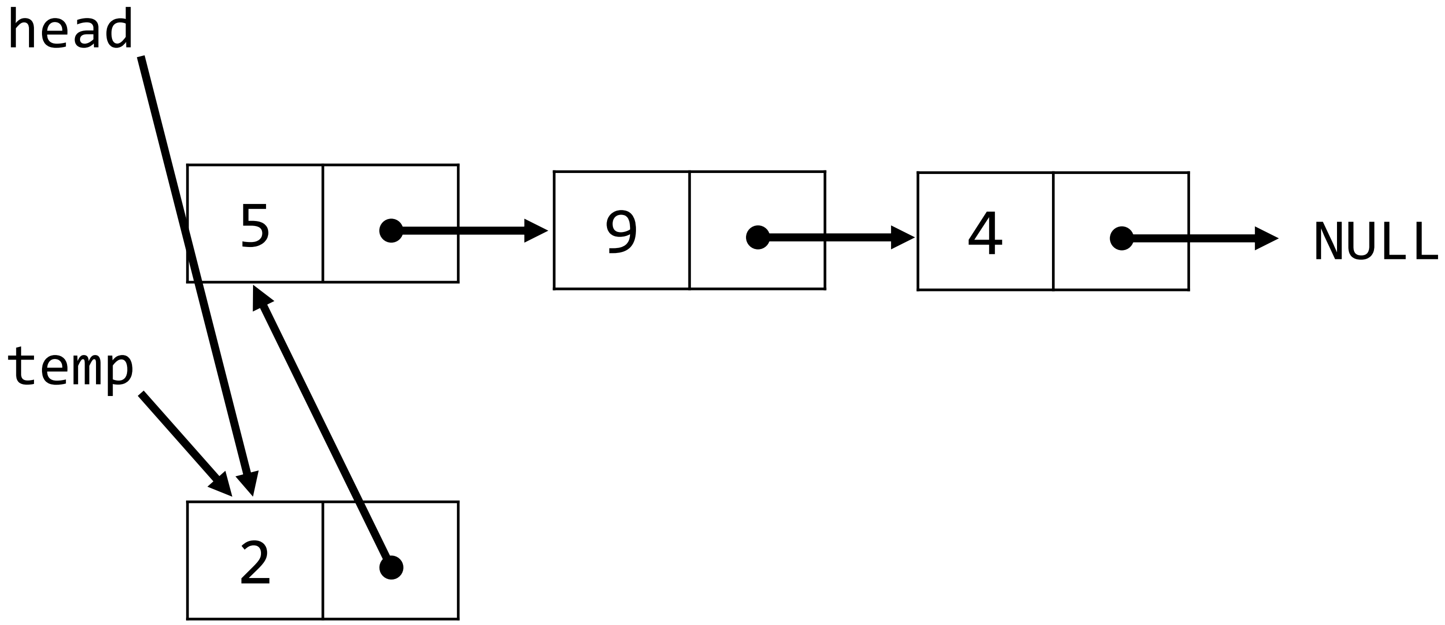
Single Linked List

push_front() operation



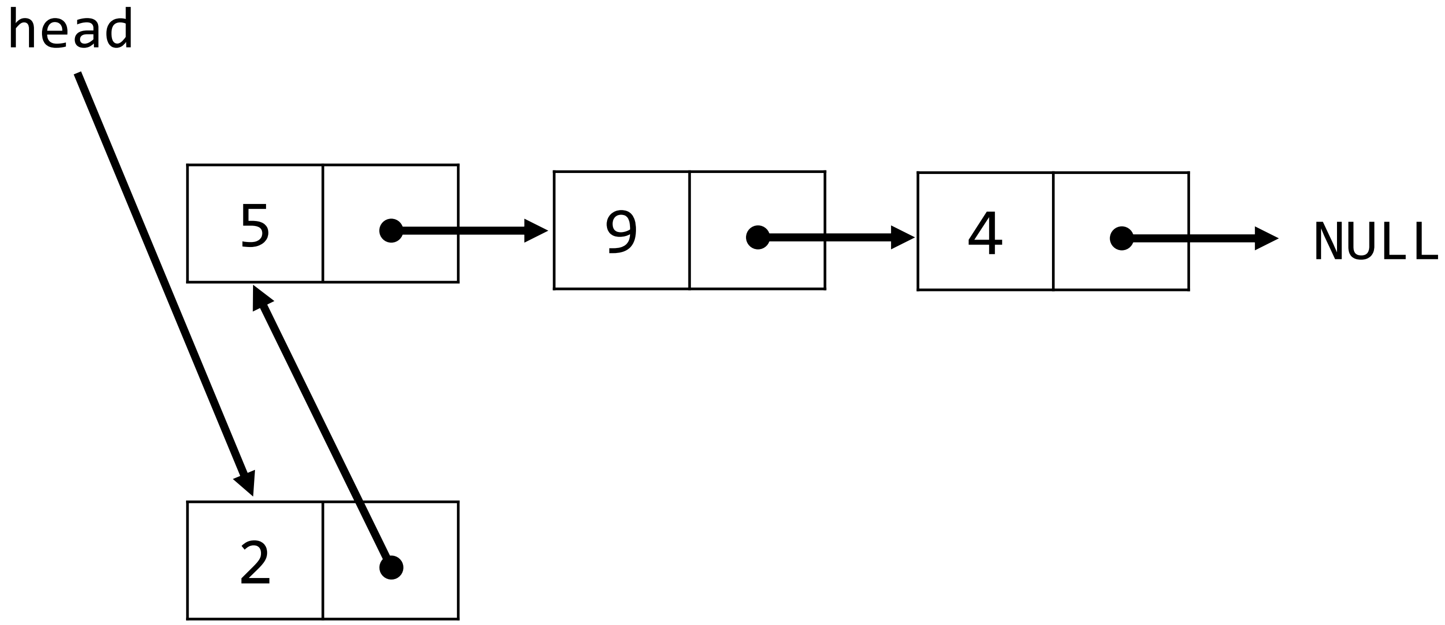
Single Linked List

push_front() operation

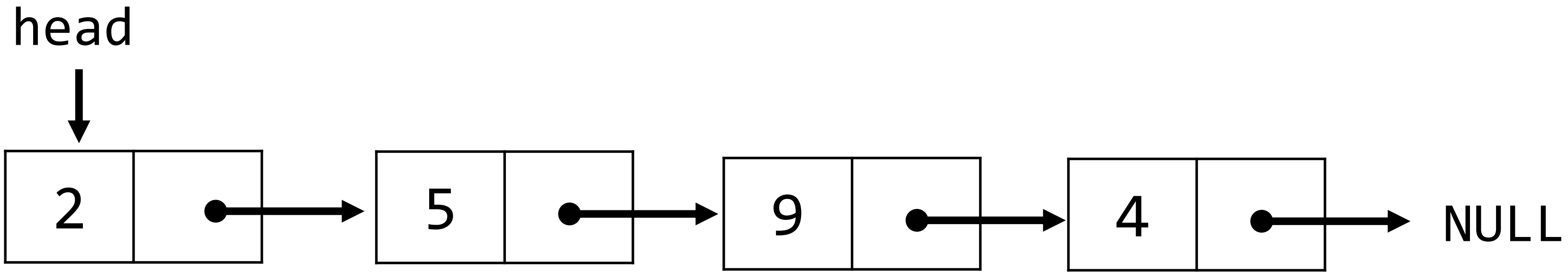


Single Linked List

push_front() operation

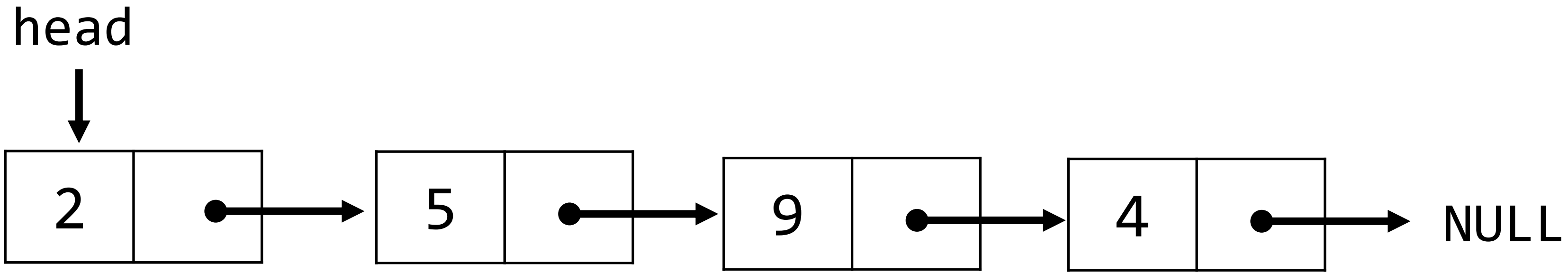


Single Linked List



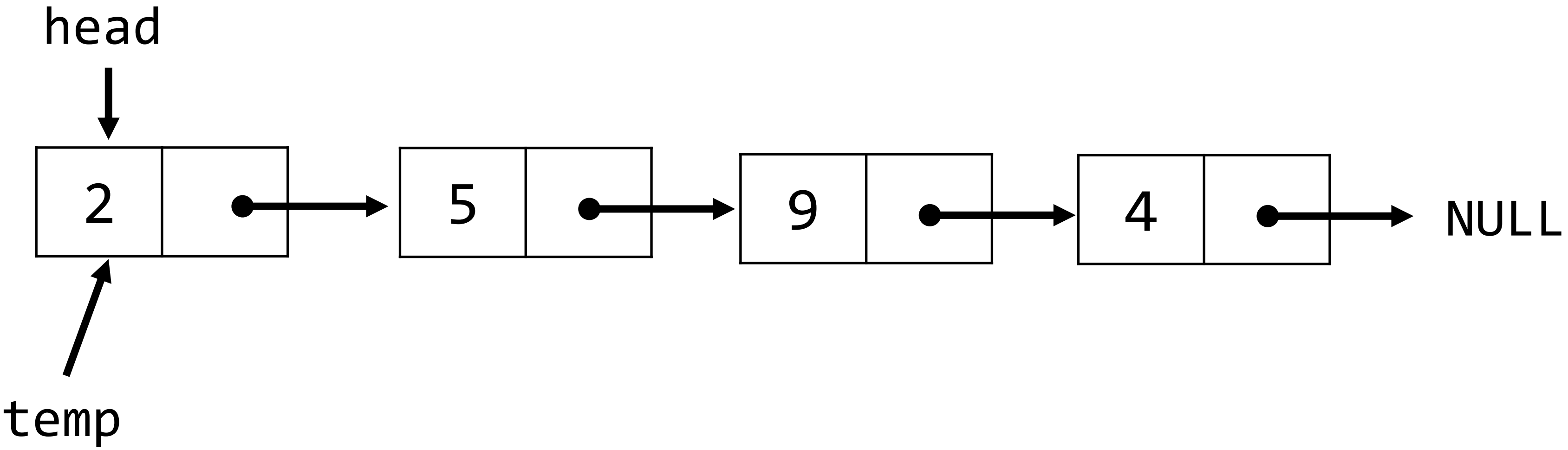
Single Linked List

pop_front() operation



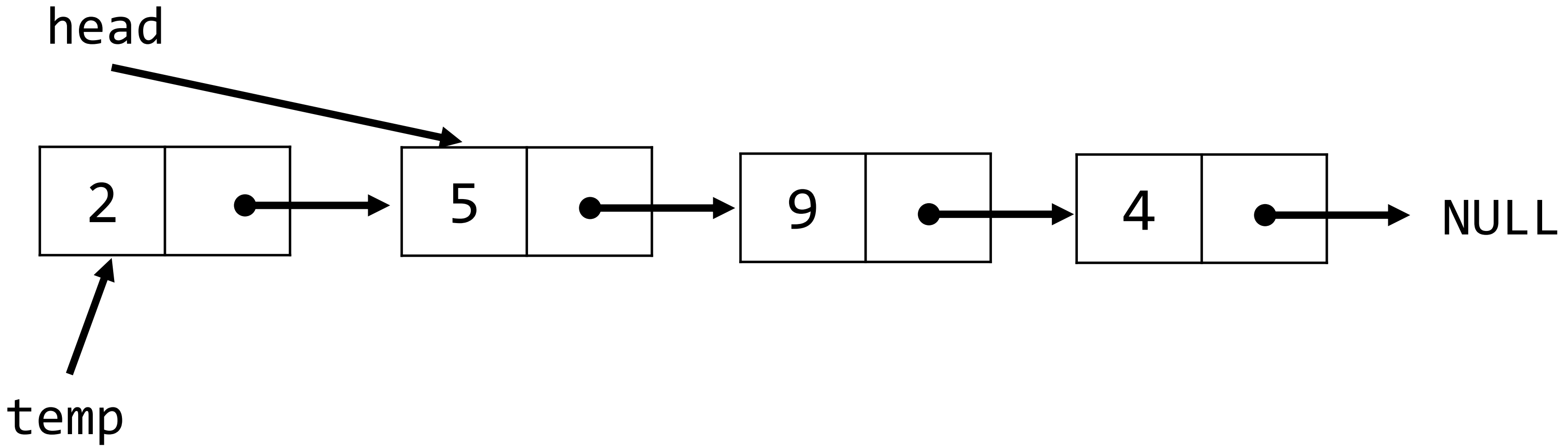
Single Linked List

pop_front() operation



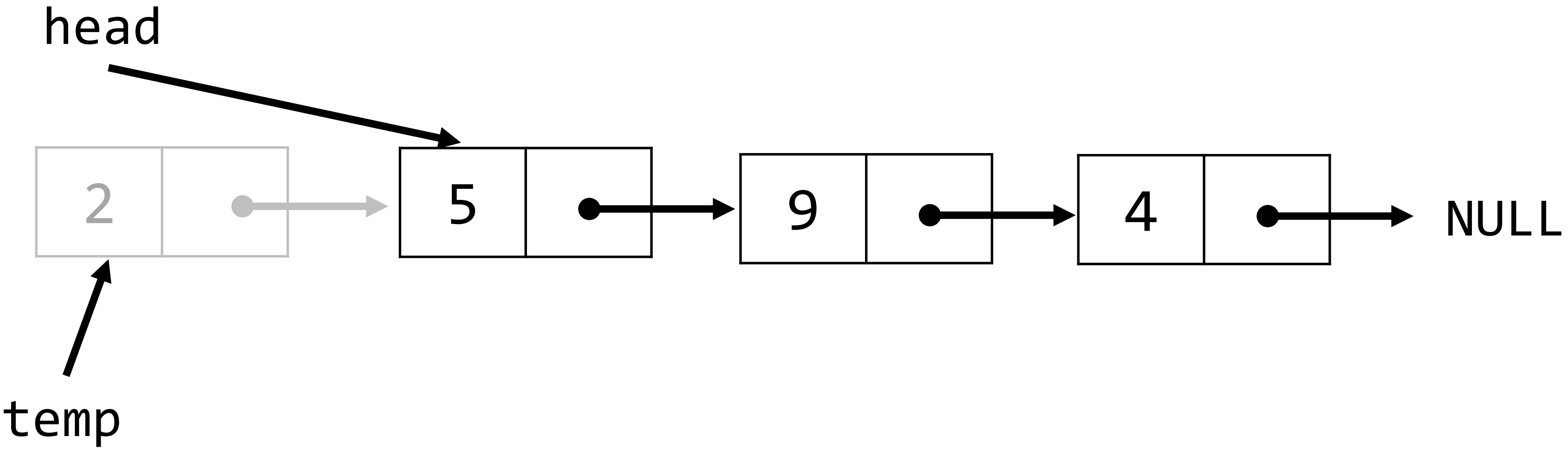
Single Linked List

pop_front() operation



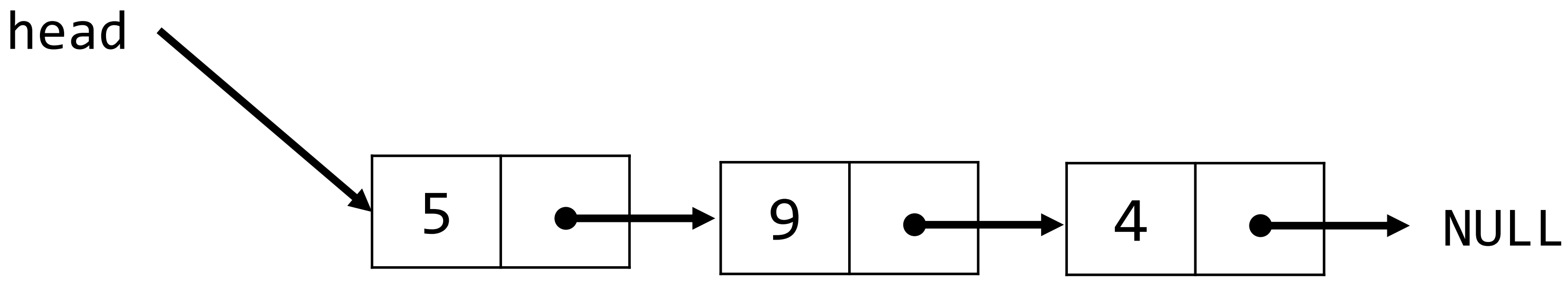
Single Linked List

pop_front() operation



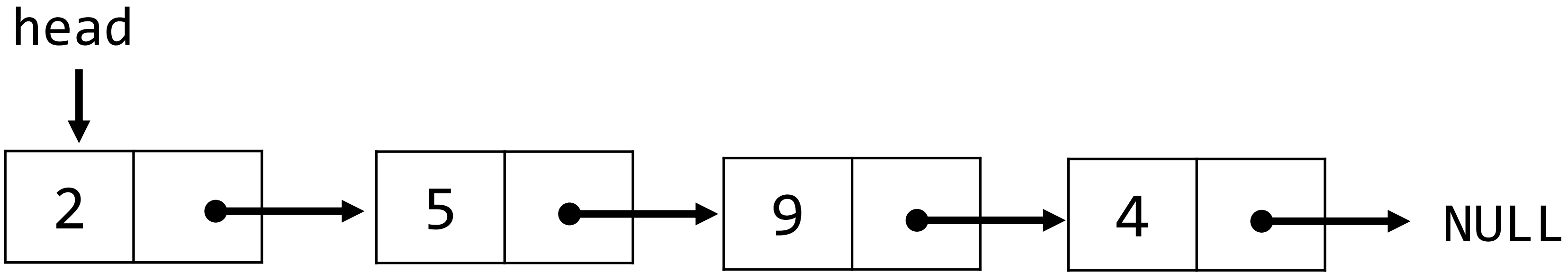
Single Linked List

pop_front() operation



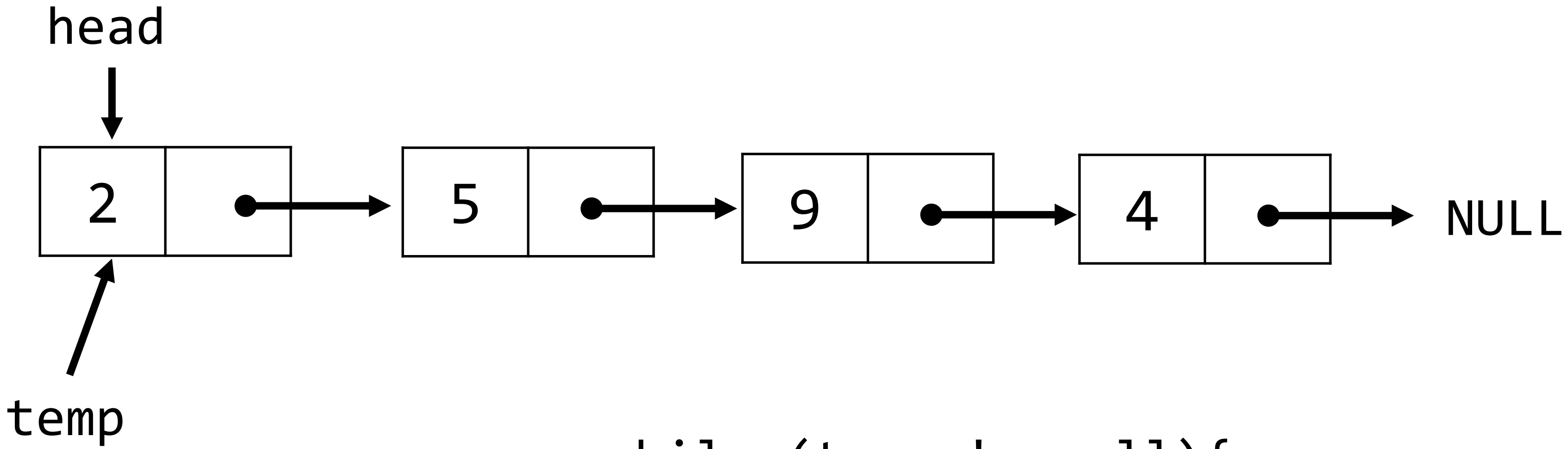
Single Linked List

search operation



Single Linked List

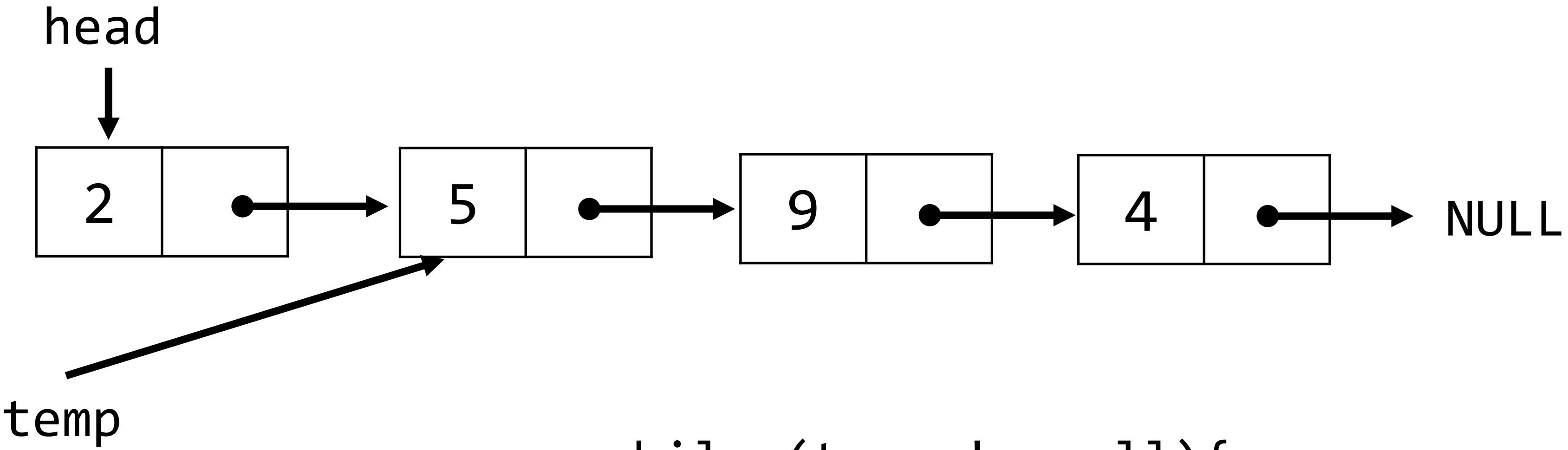
search operation : search 9



```
while (temp != null){
      if (temp->val == 9)
            return temp;
      temp = temp -> next;
}
return temp;
```

Single Linked List

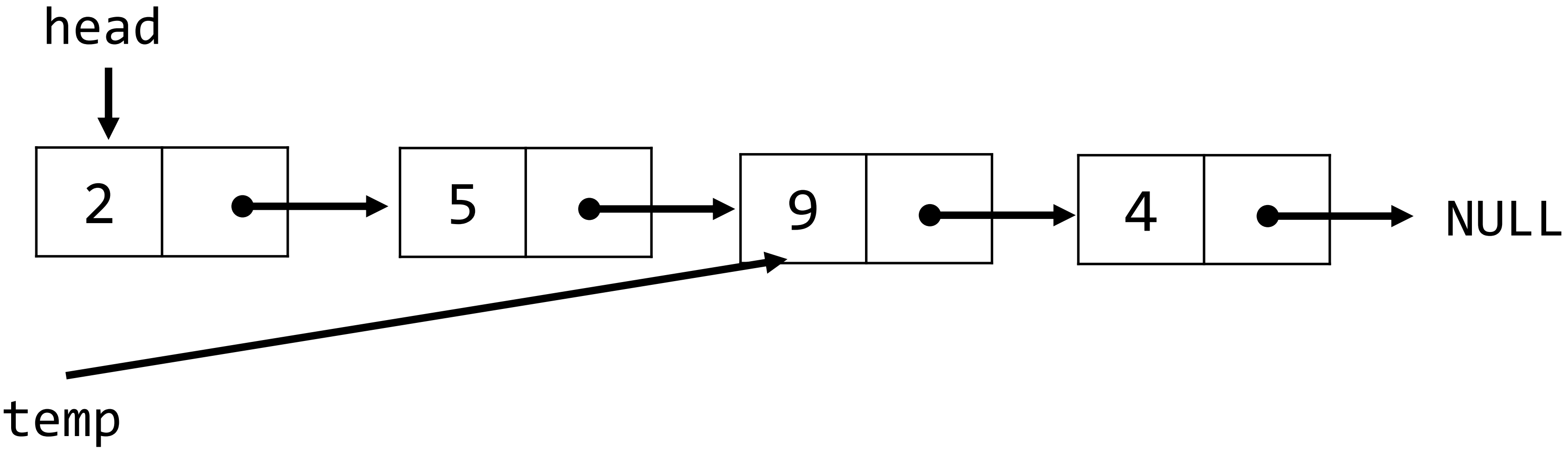
search operation : search 9



```
while (temp != null){
      if (temp->val == 9)
            return temp;
      temp = temp -> next;
}
return temp;
```

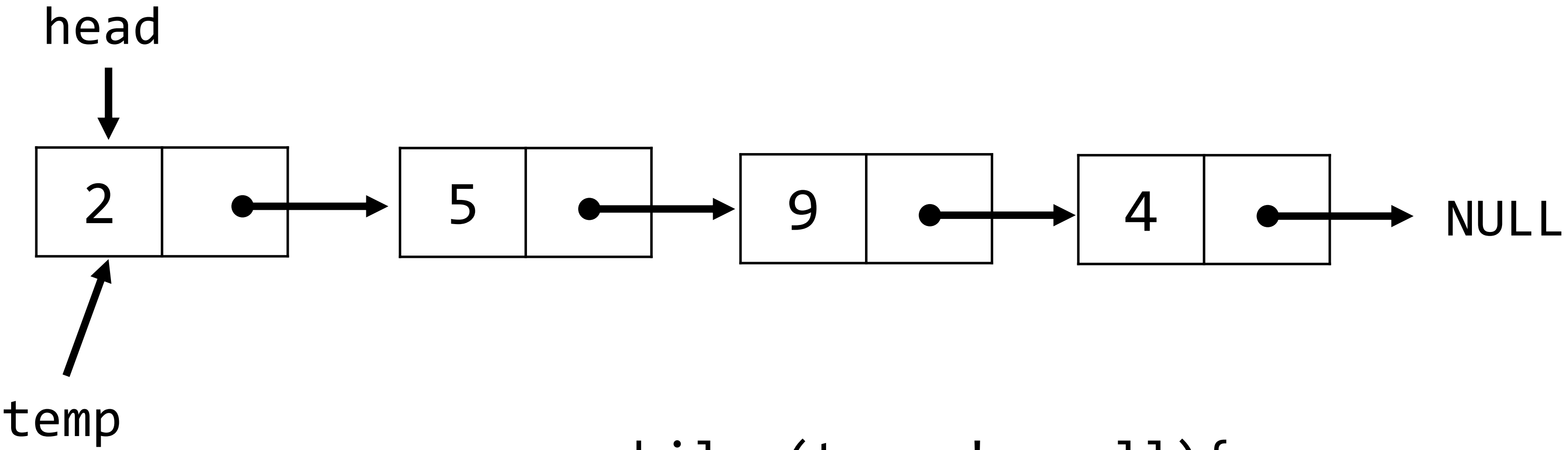
Single Linked List

search operation : search 9



Single Linked List

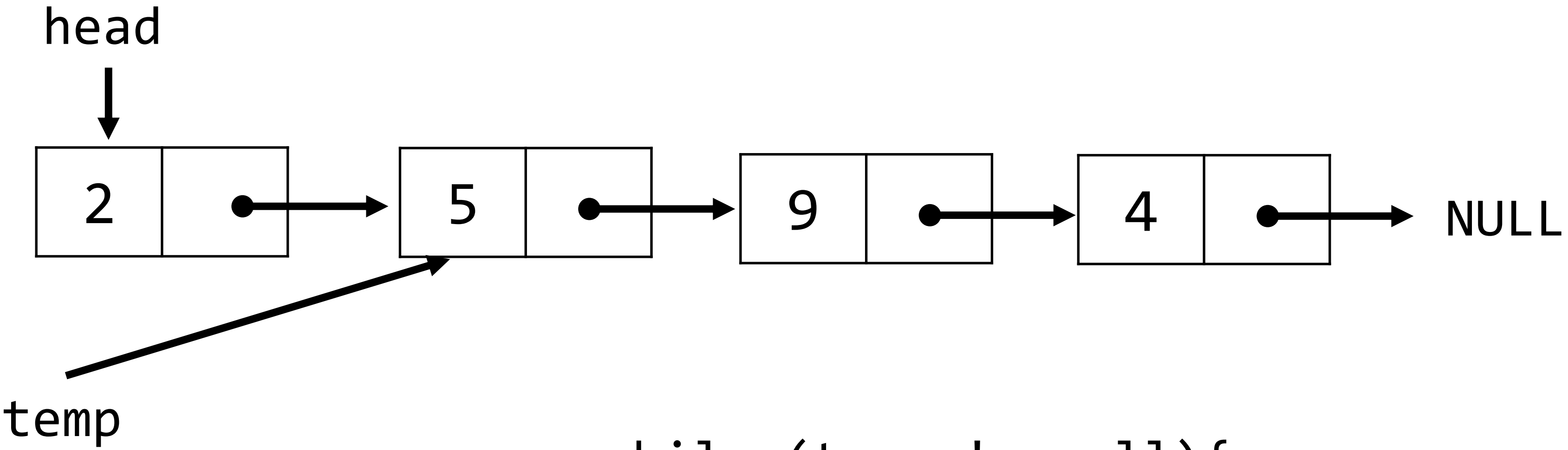
search operation : search 10



```
while (temp != null){
      if (temp->val == 10)
            return temp;
      temp = temp -> next;
}
return temp;
```

Single Linked List

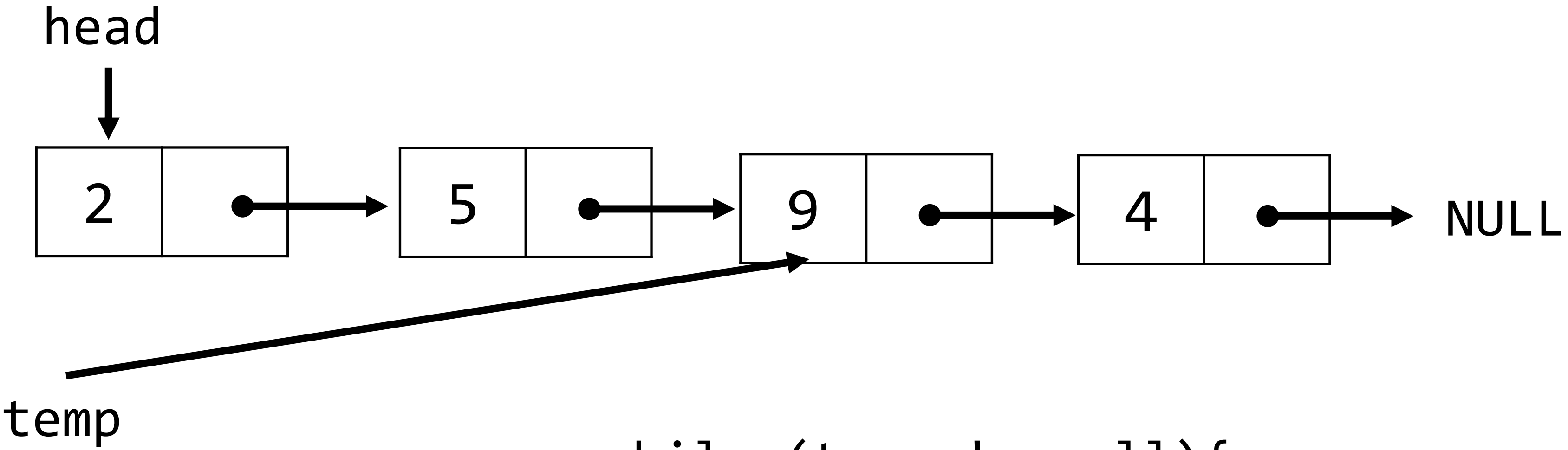
search operation : search 10



```
while (temp != null){
      if (temp->val == 10)
            return temp;
      temp = temp -> next;
}
return temp;
```

Single Linked List

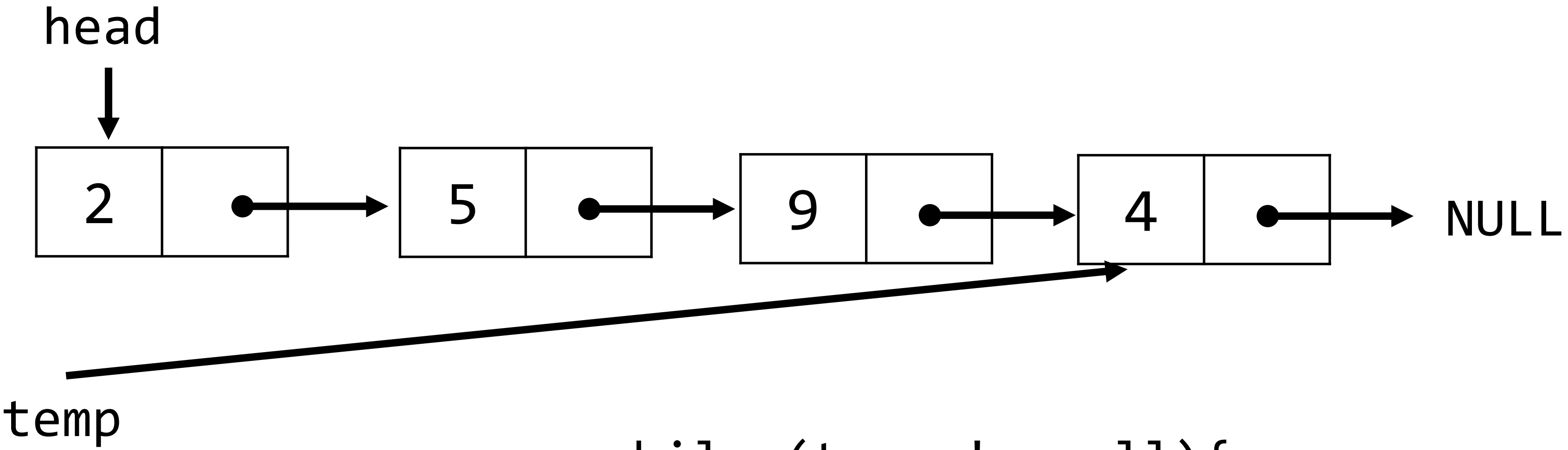
search operation : search 10



```
while (temp != null){
      if (temp->val == 10)
            return temp;
      temp = temp -> next;
}
return temp;
```

Single Linked List

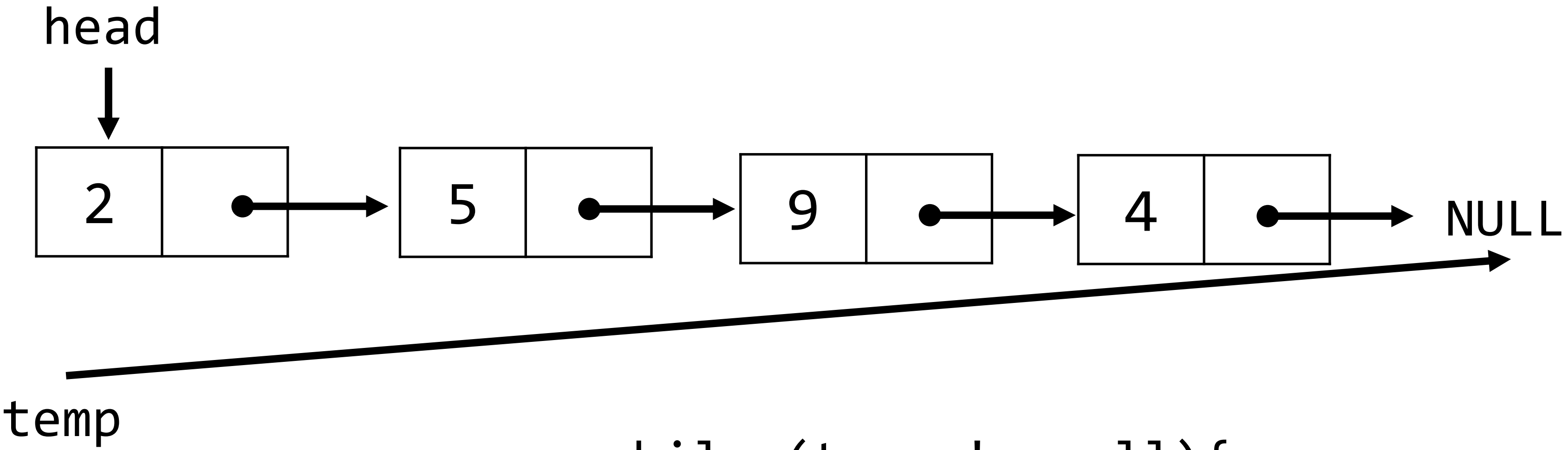
search operation : search 10



```
while (temp != null){
      if (temp->val == 10)
            return temp;
      temp = temp -> next;
}
return temp;
```

Single Linked List

search operation : search 10

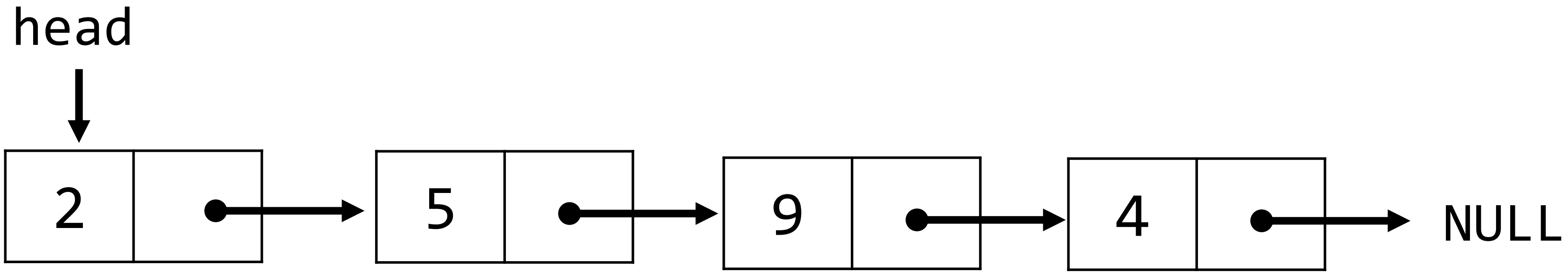


```
while (temp != null){
      if (temp->val == 10)
            return temp;
      temp = temp -> next;
}
return temp;
```

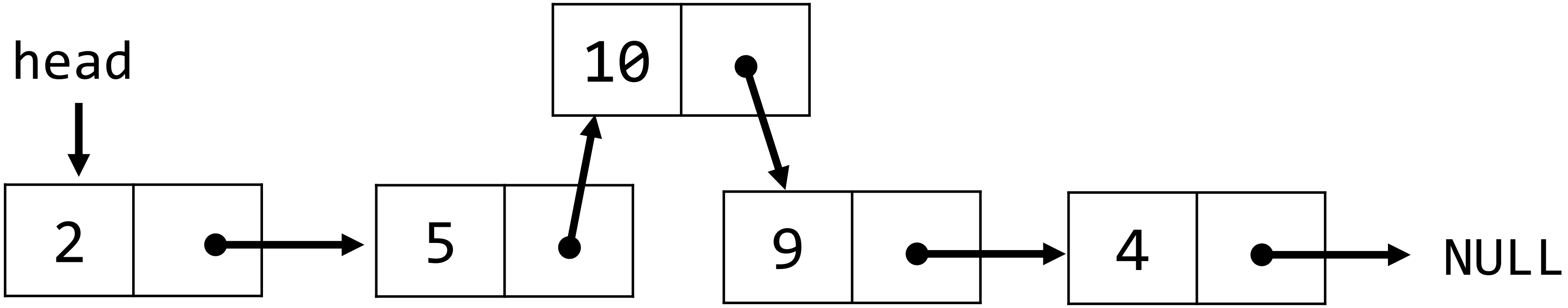
Exercise

Implement add_after(value1, value2)

- Hint: use search()



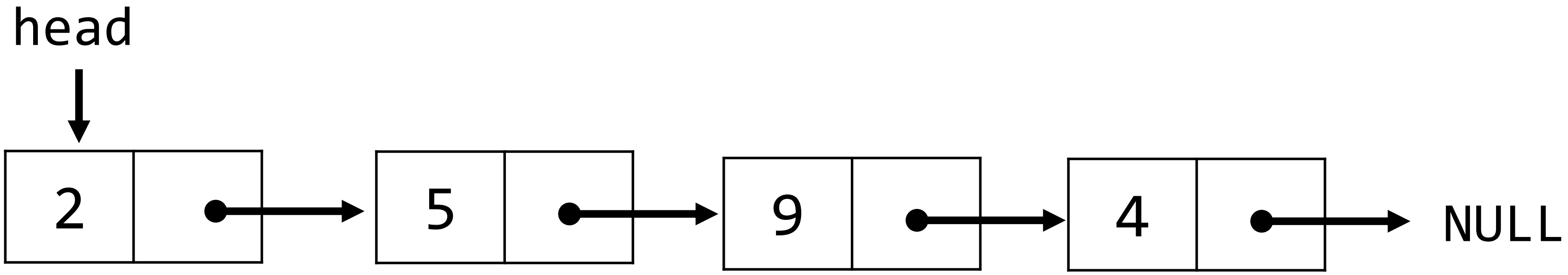
add_after(5, 10);



Delete Operation

Delete(*value*)

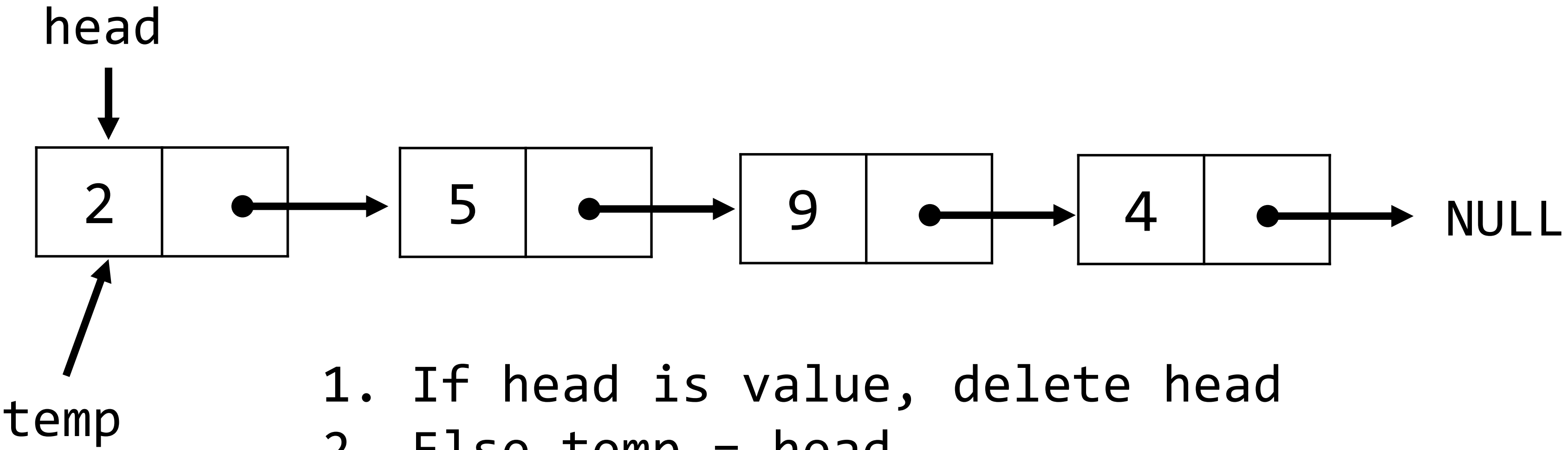
We need to find a node whose next val is *value*



Delete Operation

Delete(*value*)

We need to find a node whose next val is *value*

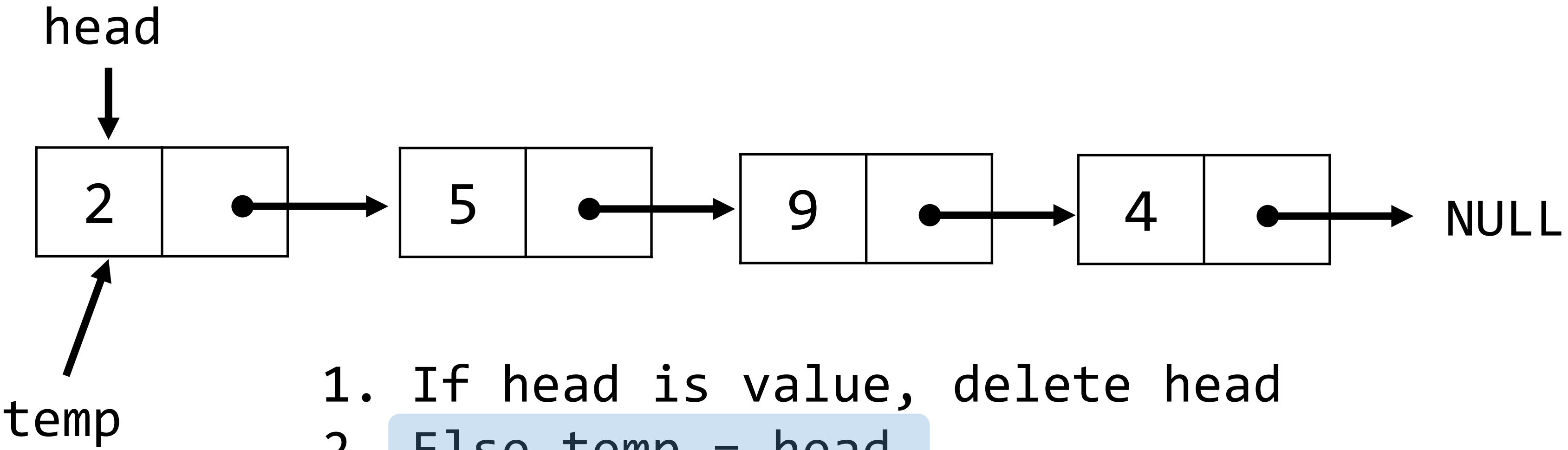


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

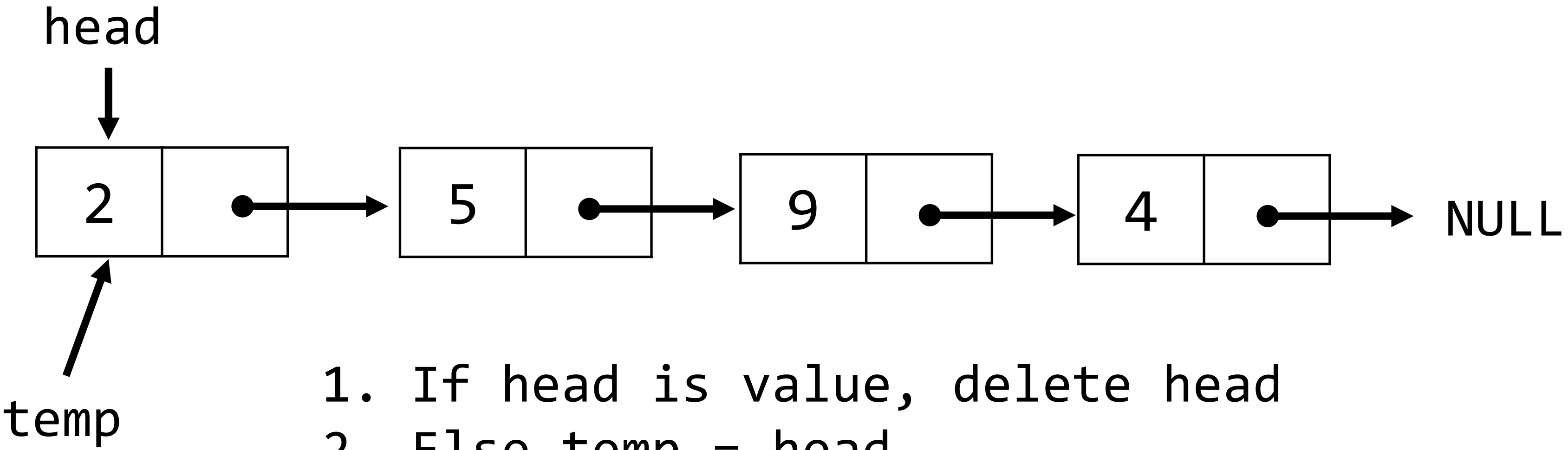


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
             temp->next= temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

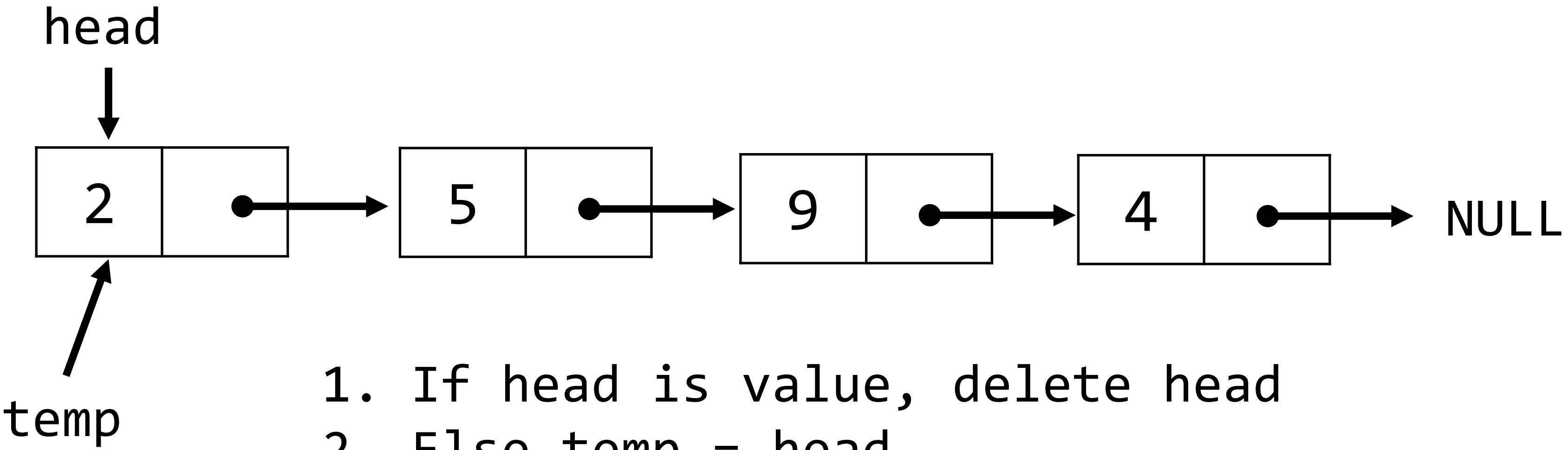


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

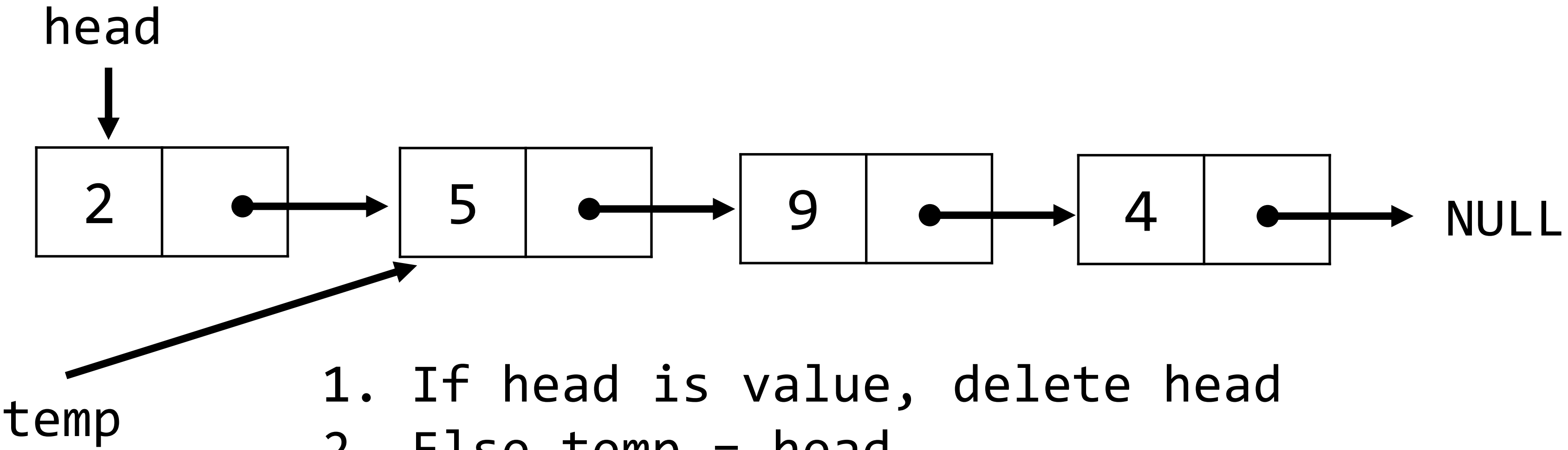


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
     If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
     temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

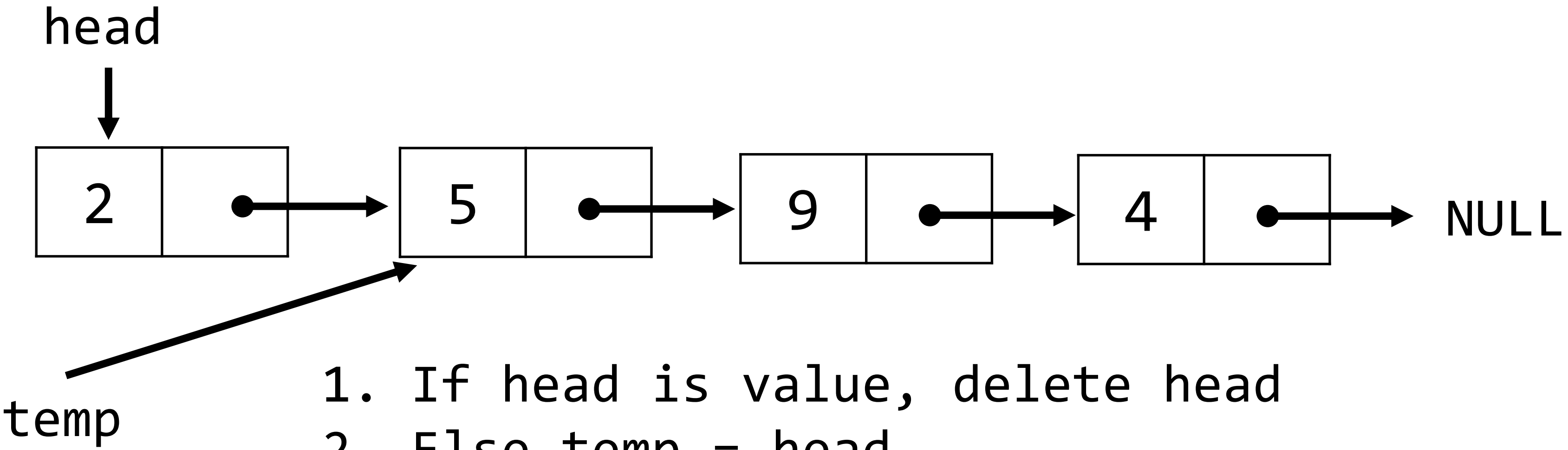


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

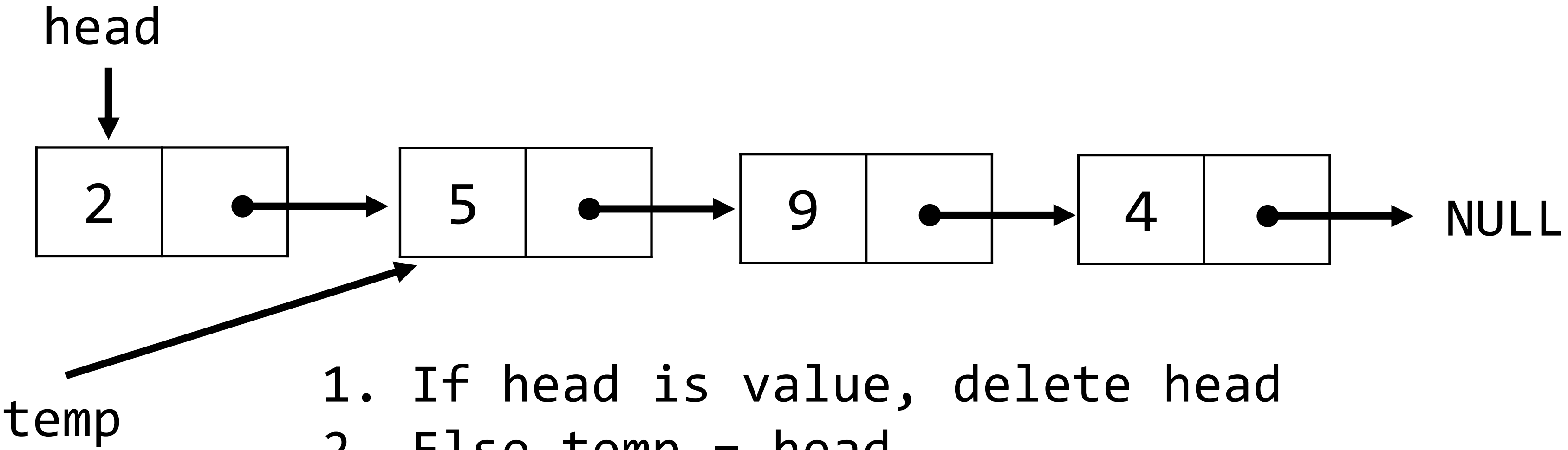


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

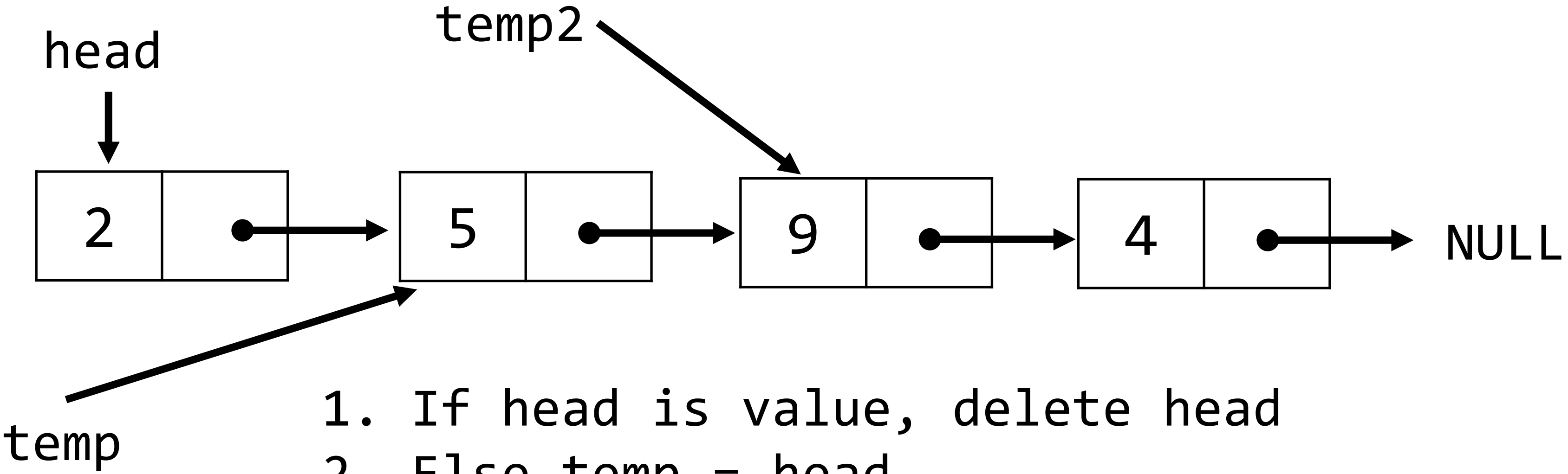


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
       If (temp->next->val == value)
             keep temp2 = temp->next
             temp->next = temp->next->next
             delete temp2
       temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

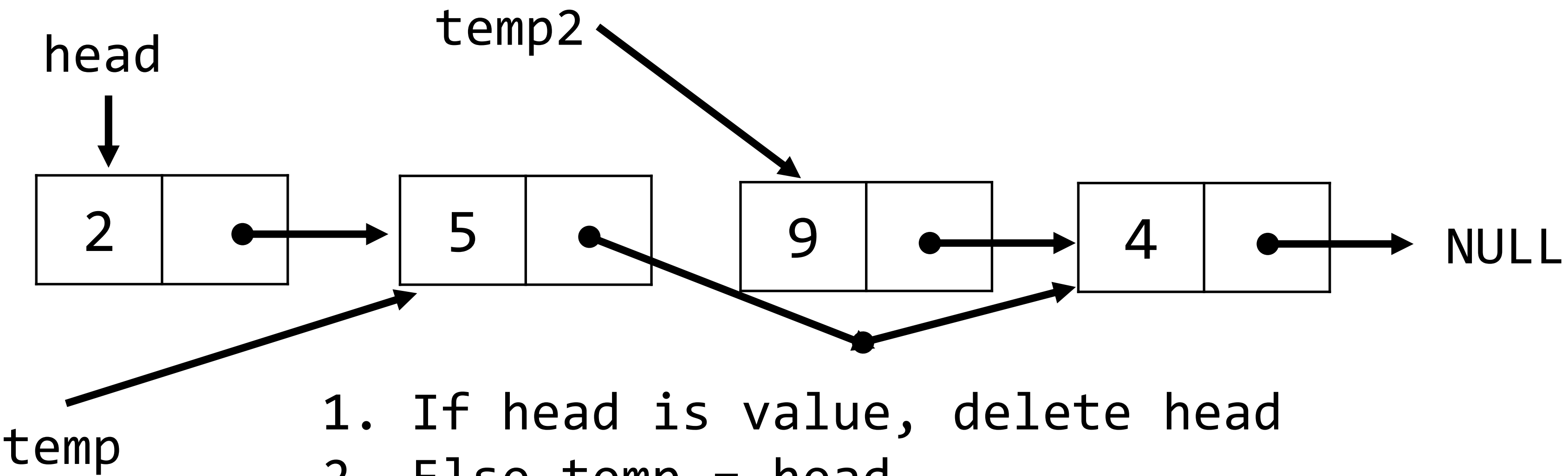


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

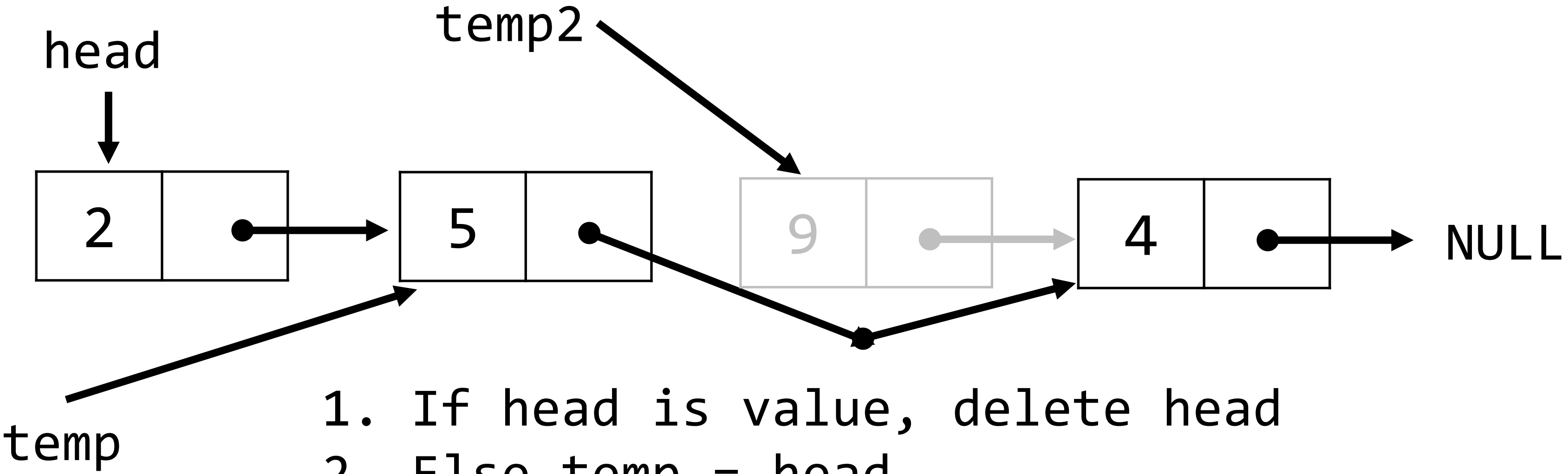


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

delete (9)

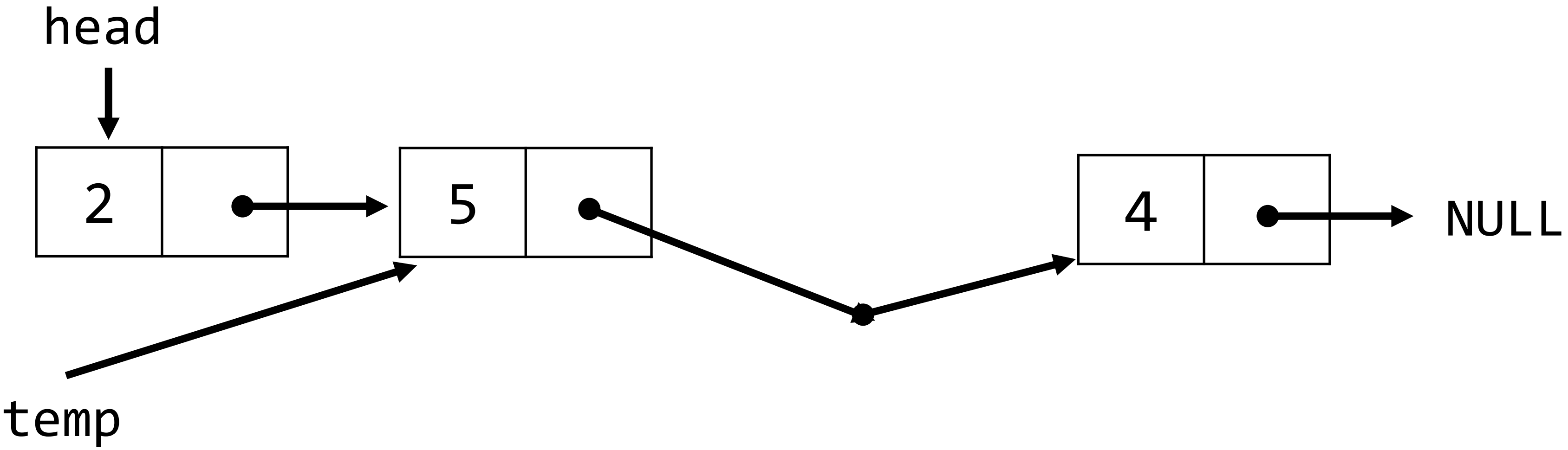


```
1. If head is value, delete head
2. Else temp = head
   While(temp ->next != NULL)
      If (temp->next->val == value)
            keep temp2 = temp->next
            temp->next = temp->next->next
            delete temp2
      temp = temp -> next
```

Delete Operation

Delete(*value*)

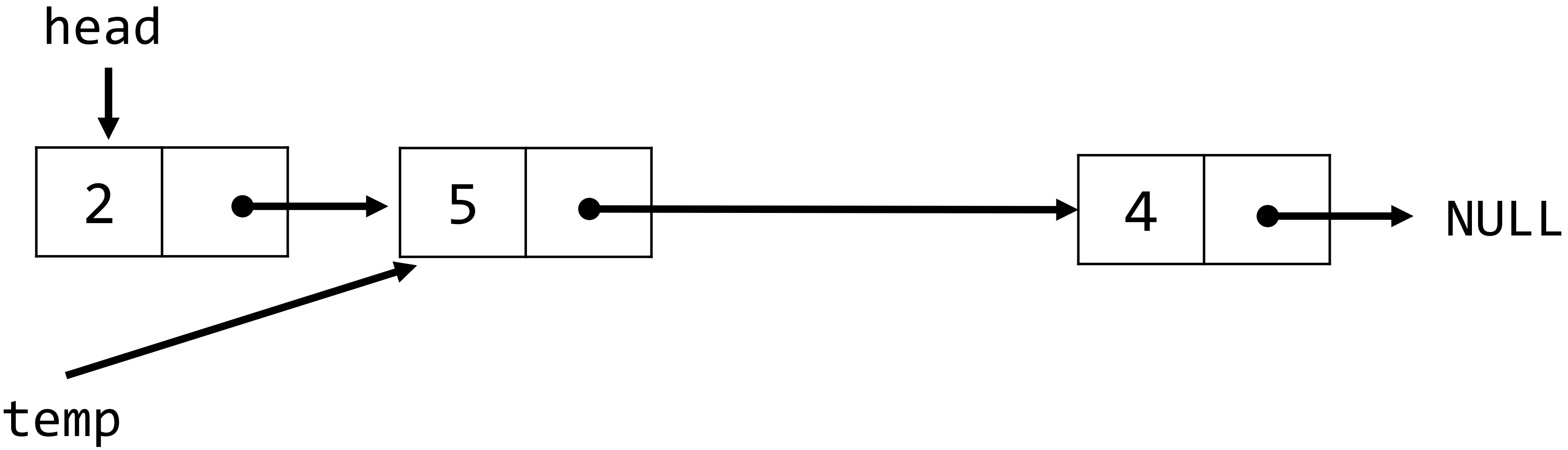
delete (9)



Delete Operation

Delete(*value*)

delete (9)



Complexity of Standard Operations

Single Linked List

Operation	**Complexity**
Insertion	
Deletion	
Display	
Search	
Delete from middle	
Clear All	
Size	
Add_after	

Source: https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm

Complexity of Standard Operations

Single Linked List

Operation	Complexity
Push_front	O(1)
Pop_front	O(1)
Display	O(n)
Search	O(n)
Delete from middle	O(n), O(1)
Clear All	O(n)
Size	O(1)
Add_after	O(n), O(1)

Source: https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm