Spring 2019 (Course Number 009952)

Data Structures and Lab
(Lecture 05: Doubly Linked List)

Prof. A. P. Shrestha, Ph.D.
Dept. of Computer Science and Engineering, Sejong University

Last Class

- Singly linked list operations
  -Insertion, Deletion

Today

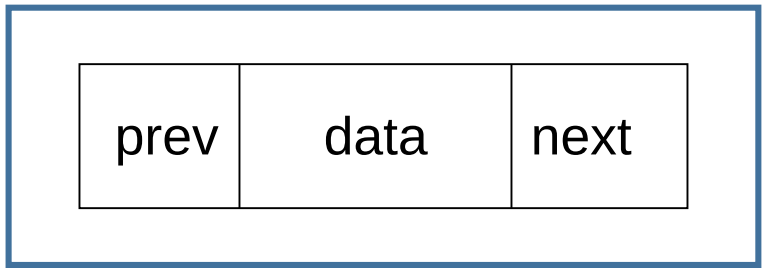
- Doubly Linked List

Next class

- XOR and Circular linked list
- Josephus Problem

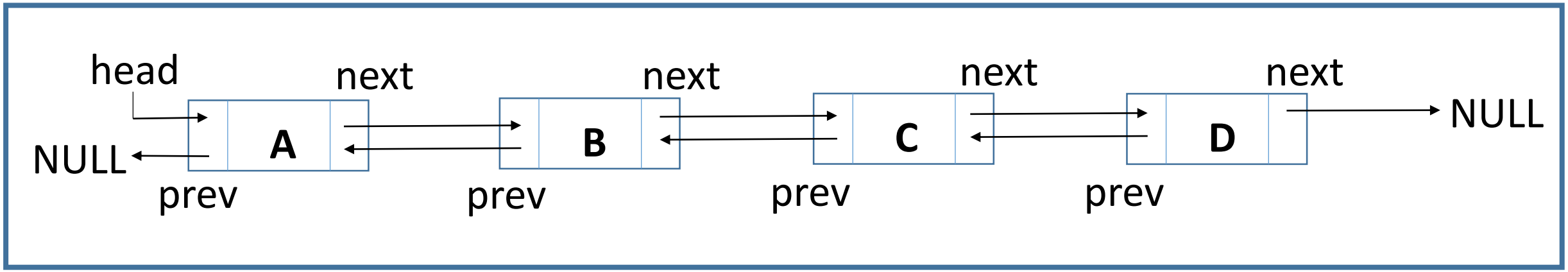
5.1.1 What's Wrong with Singly Linked List

- Moving forward in a singly-linked list is easy; but moving backwards is not so easy.
- To move back one node, we have to start at the head of the singly-linked list and move forward until the node before the current
  - A node in a singly linked list cannot be removed unless we have the pointer to its predecessor.
- To avoid this we can use *two* pointers in a node: one to point to next node and another to point to the previous node:



5.1.2 Doubly Linked List

- A Doubly Linked List (DLL), also called two-way linked list, is a variation of singly linked list
- Navigation is possible in both ways i.e. forward and backward
- It contains an extra pointer called previous pointer, together with next pointer and data



Note: The previous pointer of the first node is pointing NULL

5.1.3 Doubly Linked List - Node Implementation

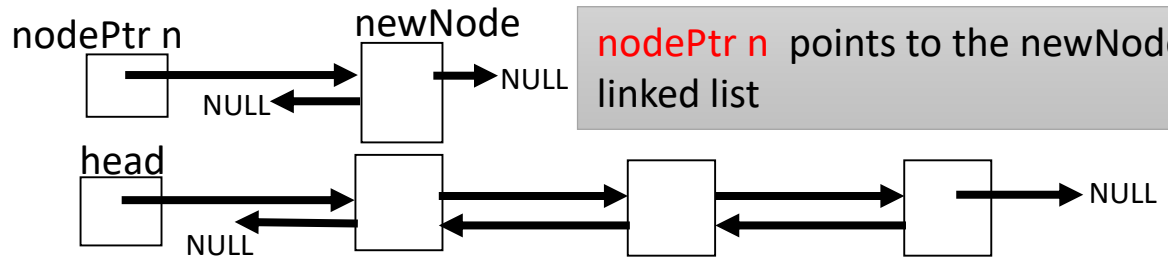
```
struct node
        {
                int data;
                node *next;   //pointer to next node
                node *prev;   //pointer to previous node
        };
```

5.2.1 Doubly Linked List- Insertion

- Insertion into a doubly-linked list has three cases (same as singly linked list):
  - Inserting a new node before the head (at the first of the list).
  - Inserting a new node after the tail (at the end of the list).
  - Inserting a new node at the middle of the list (at a particular position).

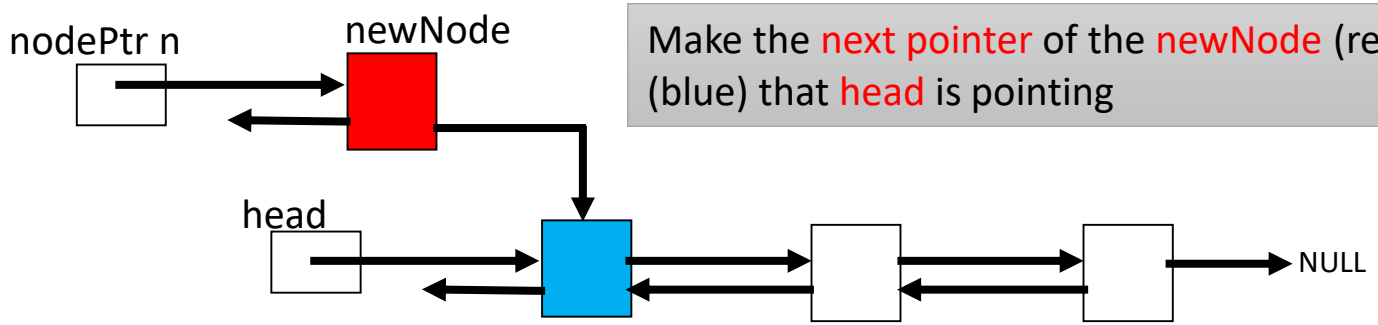
5.2.2 Inserting a Node at the Beginning

Step 1



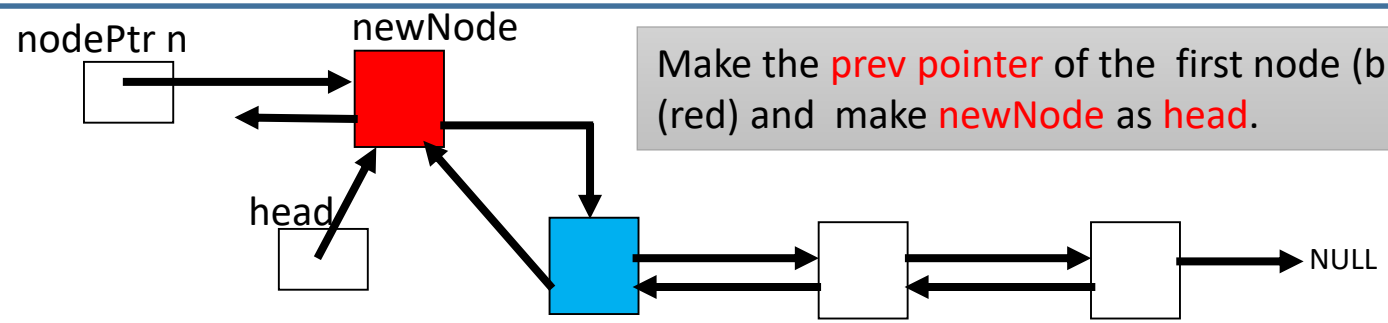
nodePtr n points to the newNode, head points to the first node of the linked list

Step 2



Make the next pointer of the newNode (red) point to the first node (blue) that head is pointing

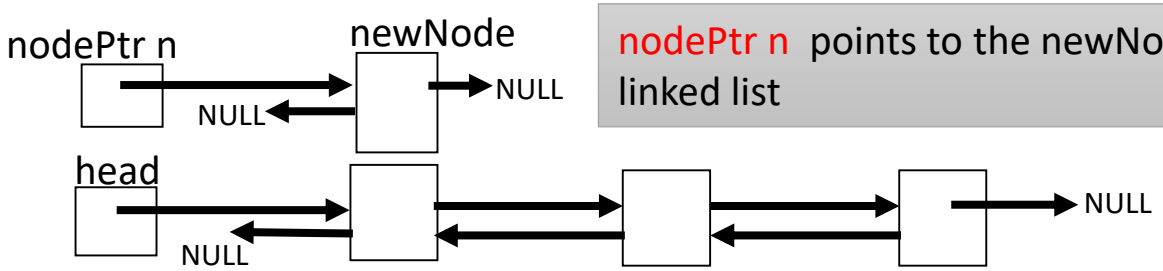
Step 3



Make the prev pointer of the first node (blue) point to the newNode (red) and make newNode as head.

5.2.3 Inserting a Node at the End

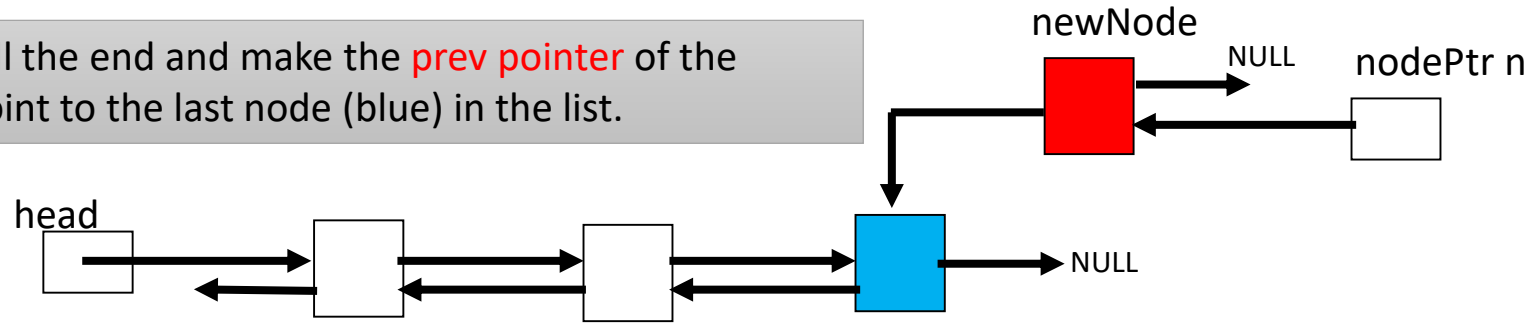
Step 1



nodePtr n points to the newNode, head points to the first node of the linked list

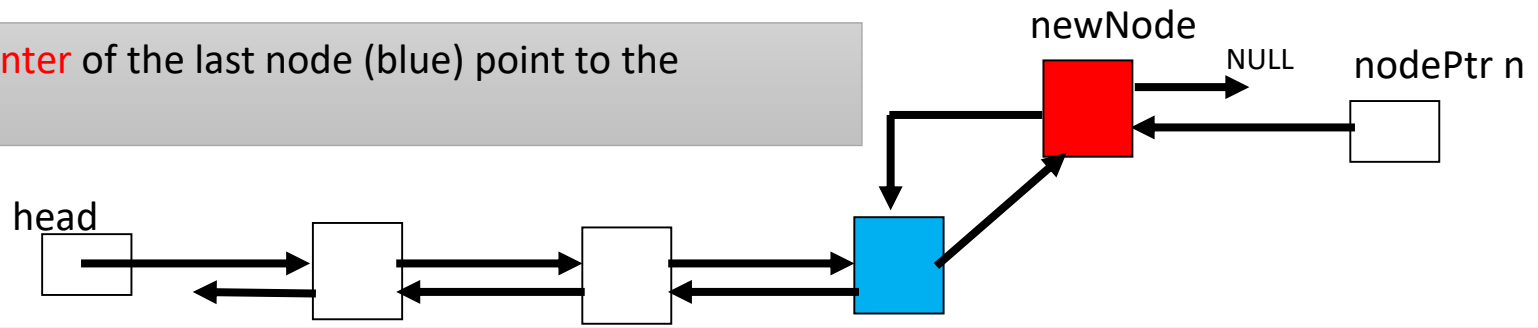
Step 2

Traverse the list till the end and make the prev pointer of the newNode (red) point to the last node (blue) in the list.



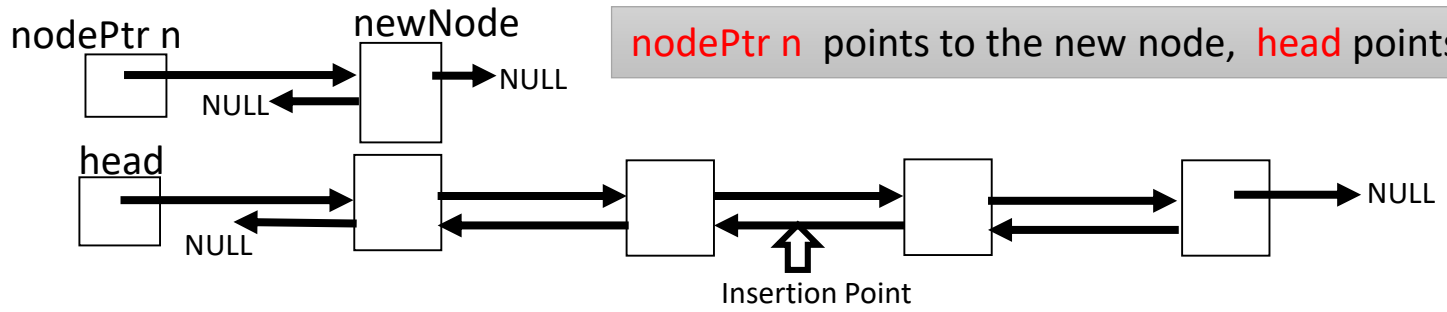
Step 3

Make the next pointer of the last node (blue) point to the newNode (red)



5.2.4 Inserting a Node at a Particular Position

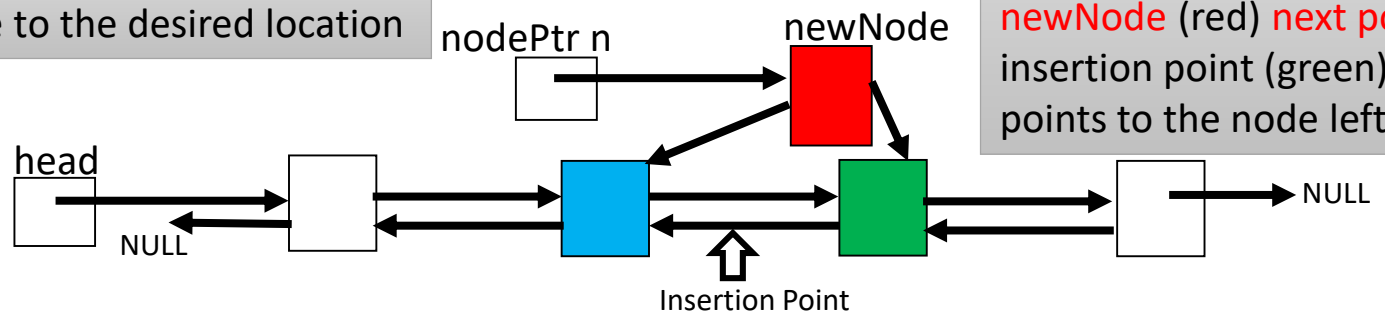
Step 1



nodePtr n points to the new node, head points to the first node of the linked list

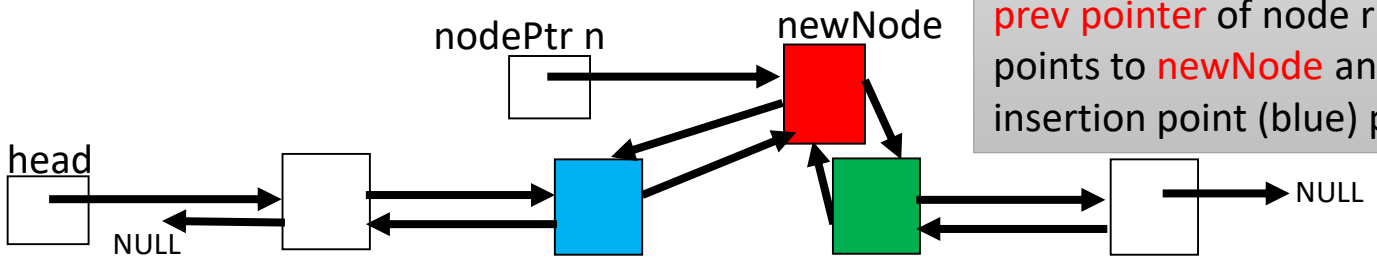
Step 2

Traverse to the desired location



newNode (red) next pointer points to the node right of the insertion point (green) and newNode (red) prev pointer points to the node left of the insertion point (blue).

Step 3



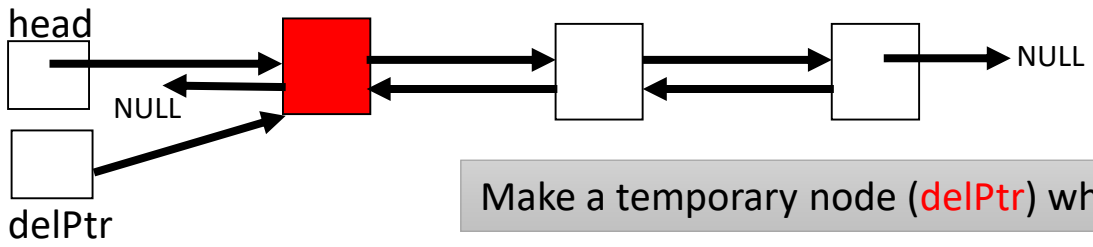
prev pointer of node right of the insertion point (green) points to newNode and next pointer of the node left of the insertion point (blue) points to newNode.

5.3.1 Doubly Linked List - Deletion

- Similar to singly linked list deletion, here we have three cases:
  - Deleting the first node
  - Deleting the last node
  - Deleting an intermediate node

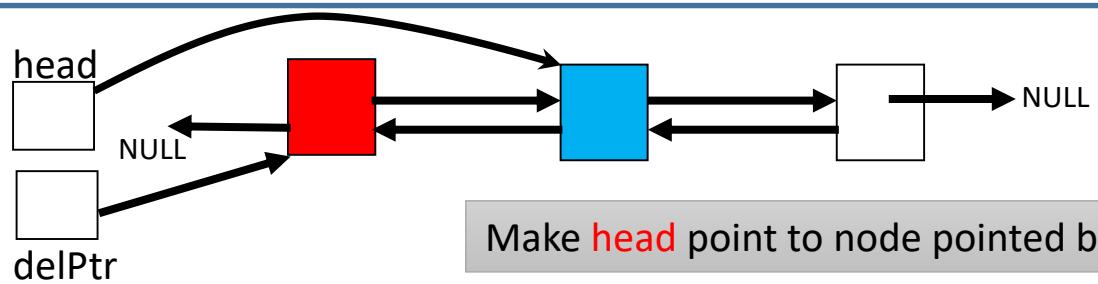
5.3.2 Deleting the First Node

Step 1



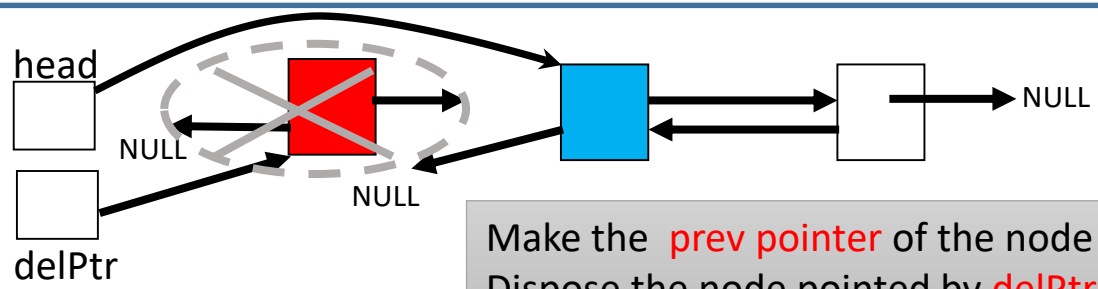
Make a temporary node (delPtr) which will point to the same node as that of head

Step 2



Make head point to node pointed by next pointer of the first node.

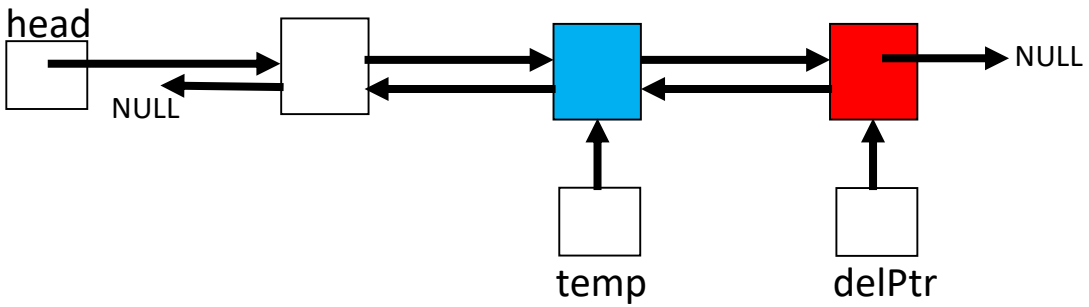
Step 3



Make the prev pointer of the node (blue) pointed by head to NULL
Dispose the node pointed by delPtr

5.3.3 Deleting the Last Node

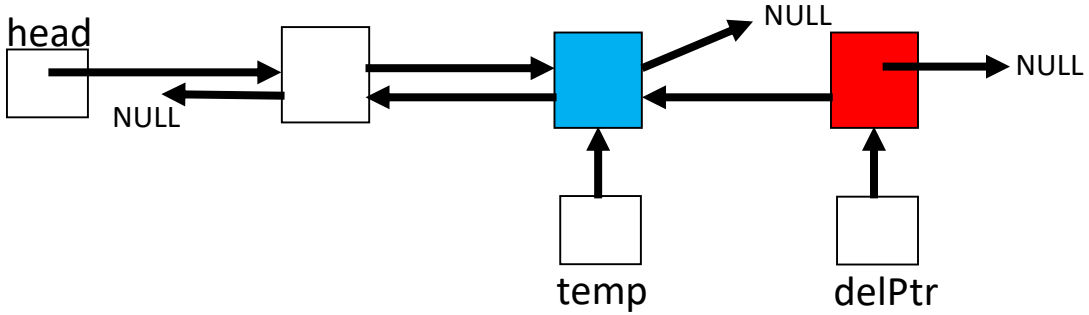
Step 1



Traverse the list till the end with delPtr while maintaining the previous node address in temp pointer

temp=delPtr->prev

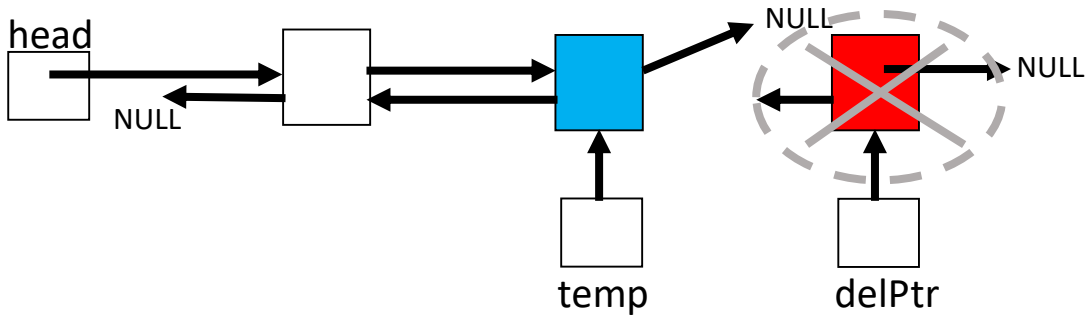
Step 2



Make the next pointer of the node pointed by temp to NULL

temp->next=NULL

Step 3

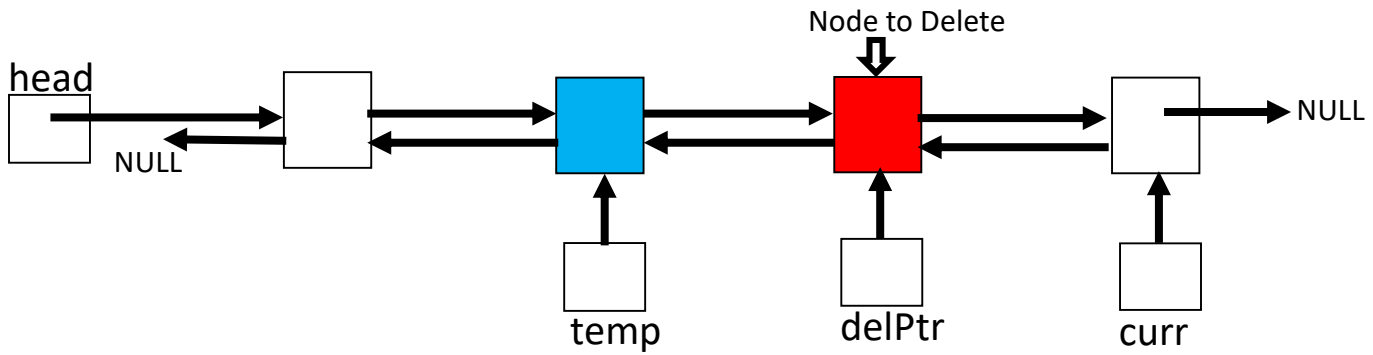


Dispose of the node pointed by delPtr

delete(delPtr)

5.3.4 Deleting an Intermediate Node

Step 1

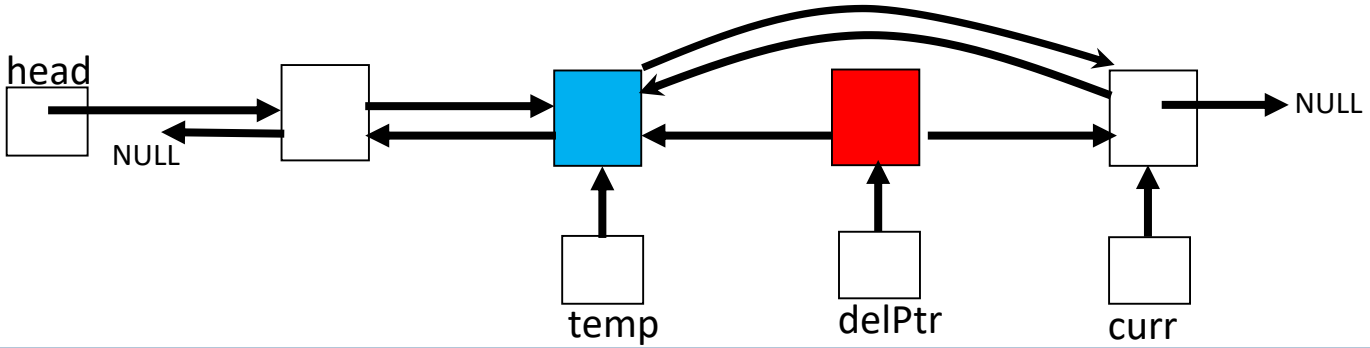


Traverse the list so that delPtr, temp, and curr points corresponding node as shown

curr = delPtr->next

temp = delPtr->prev

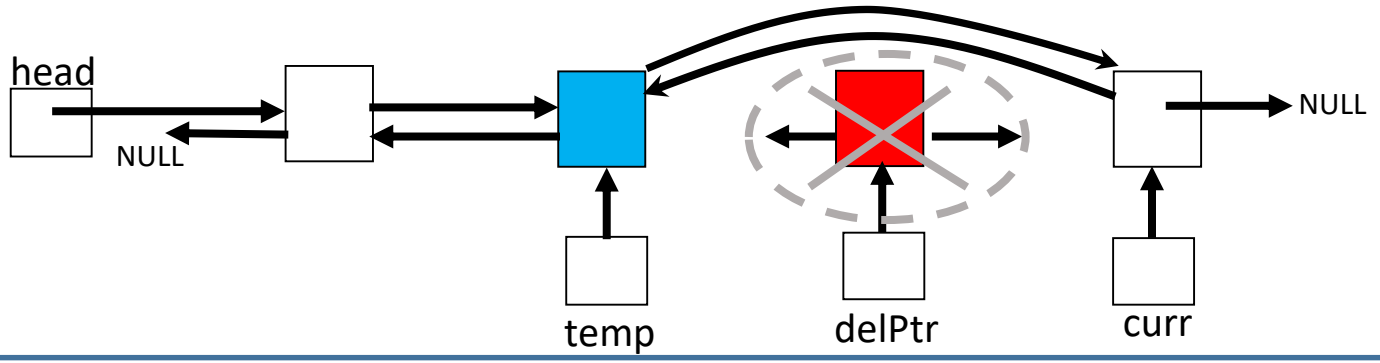
Step 2



temp->next = curr

curr->prev= temp

Step 3



Dispose the node pointed by delPtr

Q & A ?