Chapter 5 Linked List

ผศ.ดร.สิลดา อินทรโสธรฉันท์

สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยขอนแก่น

Linked List

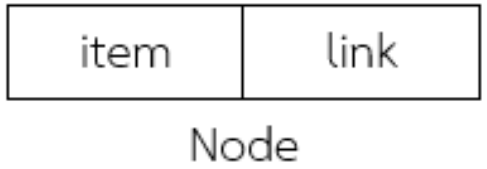
- การเก็บข้อมูลแบบอาเรย์ต้องกำหนดขนาดหรือจองพื้นที่หน่วยความจำก่อนใช้งานและไม่สามารถเพิ่มหรือลดขนาดพื้นที่ในการจัดเก็บได้อัตโนมัติ
- การเก็บข้อมูลที่เรียกว่าลิงค์ลิสต์ ซึ่งสามารถเพิ่มลดขนาดได้แบบอัตโนมัติ
- หลักการทำงานของลิงค์ลิสต์จะใช้การจัดสรรหน่วยความจำแบบพลวัต คือ เมื่อมีสมาชิกใหม่เข้ามาจึงจะทำการจัดสรรเนื้อที่ให้สำหรับเก็บข้อมูลนั้น ทำให้ตำแหน่งของสมาชิกแต่ละตัวไม่อยู่ต่อเนื่องกัน

Linked List

- ในข้อมูลชุดเดียวกันจะทำการเก็บข้อมูลตำแหน่งของสมาชิกในลักษณะที่เชื่อมโยงต่อกันไป
- สมาชิกแต่ละตัวของลิงค์ลิสต์จะเรียกว่าโหนด (node) โดยแต่ละโหนดจะประกอบด้วยข้อมูล 2 ส่วน คือ ส่วนเก็บข้อมูล (item) และส่วนที่เก็บที่อยู่ของข้อมูลในตำแหน่งถัดไปเรียกว่า ลิงค์ (link)
- โครงสร้างลิงค์ลิสต์แบ่งเป็น 2 ประเภท คือ ลิงค์ลิสต์ทางเดียว (Single Linked List) กับลิงค์ลิสต์สองทาง (Doubly Linked List)

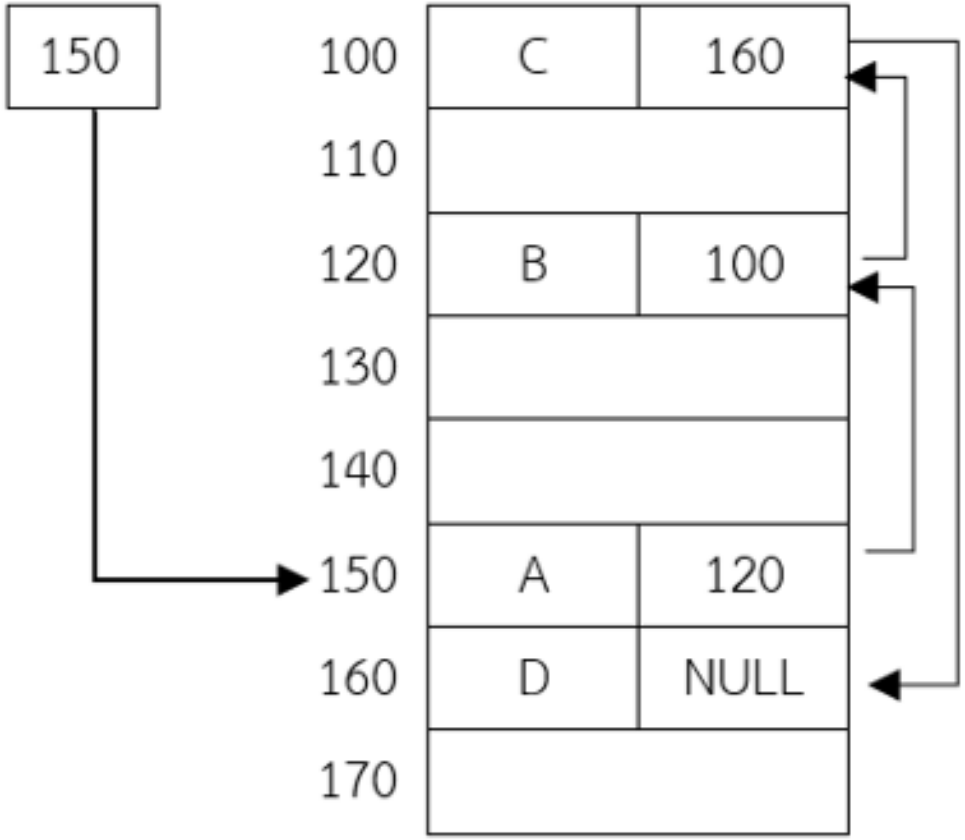
Single Linked List

- แต่ละโหนดจะประกอบด้วย 2 ส่วน คือ ส่วนเก็บข้อมูล (item) และส่วนของลิงค์ (link)
- ลักษณะการเก็บข้อมูลในหน่วยความจำด้วยลิงค์ลิสต์ทางเดียว สมาชิกแต่ละตัวจะเก็บตำแหน่งของข้อมูลตัวต่อไปไว้เชื่อมโยงกันไป
- โดยสมาชิกตัวสุดท้ายจะมีค่าในส่วนของลิงค์เป็น NULL เพื่อเป็นการบอกว่า ไม่มีข้อมูลอื่นอีก



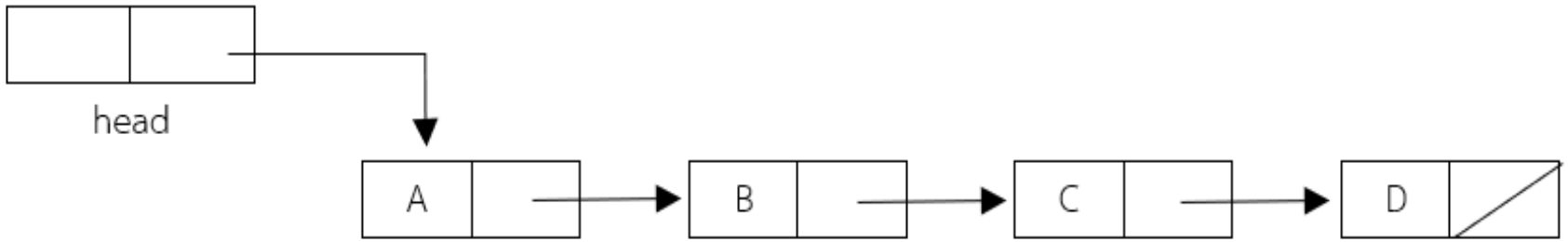
Node

Single Linked List



Single Linked List

- เพื่อให้ง่ายต่อการอธิบายการทำงานของลิงค์ลิสต์ สามารถเขียนใหม่ได้ดังนี้



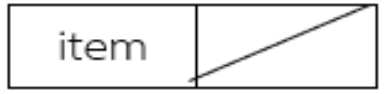
- จากภาพจะมีตัวแปร head ทำหน้าที่ในการเก็บตำแหน่งข้อมูลตัวแรกของลิงค์ลิสต์ไว้ และสมาชิกแต่ละตัวก็จะเชื่อมโยงกันตามลำดับจนถึงโหนดสุดท้ายที่มีค่าลิงค์เป็น NULL

การสร้างโหนด

- จากโครงสร้างของลิงค์ลิสต์จะพบว่า ลักษณะของโหนดจะมีความแตกต่างกันตามหน้าที่และตำแหน่งของโหนดในลิงค์ลิสต์ ซึ่งสามารถแบ่งเป็นหลักๆ 3 แบบคือ
  - โหนด head ทำหน้าที่เก็บตำแหน่งของโหนดแรกไว้ที่ส่วนของลิงค์ และจะไม่มีการเก็บค่าในส่วนข้อมูล
  - โหนดทั่วไป
  - โหนดสุดท้าย จะมีค่าของลิงค์เป็น NULL

head

โหนด



โหนดสุดท้าย

Constructor ของคลาสโหนด

- จากโครงสร้างของโหนด มาใช้ในการกำหนด attribute ได้ดังนี้

```
private Object item;

private Node link;
```

- constructor สำหรับสร้าง head node

```
public Node(){

        link = null;

 }
```

Constructor ของคลาสโหนด

- constructor สำหรับสร้าง last node

```
public Node(Object newItem){
        item = newItem;
        link = null;
}
```

Constructor ของคลาสโหนด

- constructor สำหรับสร้าง node ใดๆ

```
public Node(Object newItem, Node nextNode){
        item = newItem;
        link = nextNode;
}
```

เมธอดต่าง ๆ ของคลาสโหนด

- เมธอดในการ set และ get ค่าในส่วนของ item

```
public void setItem (Object newItem){
        item = newItem;
}
 public Object getItem(){
        return item;
 }
```

เมธอดต่าง ๆ ของคลาสโหนด

- เมธอดในการ set และ get ค่าในส่วนของ link

```
public void setLink(Node nextNode){
        link = nextNode;
}
public Node getLink(){
        return link;
}
```

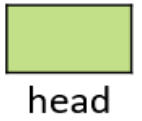
การดำเนินการบนลิงค์ลิสต์ทางเดียว

- การสร้างส่วนหัวของลิงค์ลิสต์ (การสร้างลิสต์)
- การค้นหาตำแหน่งที่ต้องการ
- การลบโหนด
- การแทรกโหนด

การสร้างส่วนหัวของลิงค์ลิสต์

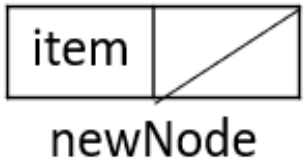
- การสร้างลิงค์ลิสต์จึงเริ่มด้วยส่วนหัวและการเพิ่มโหนด โดยมีขั้นตอนการทำงานดังนี้
- Step1: ทำการสร้างโหนด head

```
Node head = new Node();
```



- Step2: ทำการสร้างโหนดที่ต้องการเพิ่มเข้าสู่ลิงค์ลิสต์

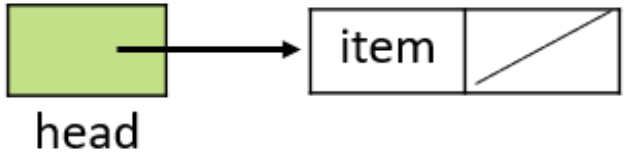
```
Node newNode = new Node(item);
```



การสร้างส่วนหัวของลิงค์ลิสต์

- Step3: ทำการกำหนดค่า head ให้ชี้ไปที่ตำแหน่งของโหนดที่เพิ่มเข้าไป

```
head = newNode;
```

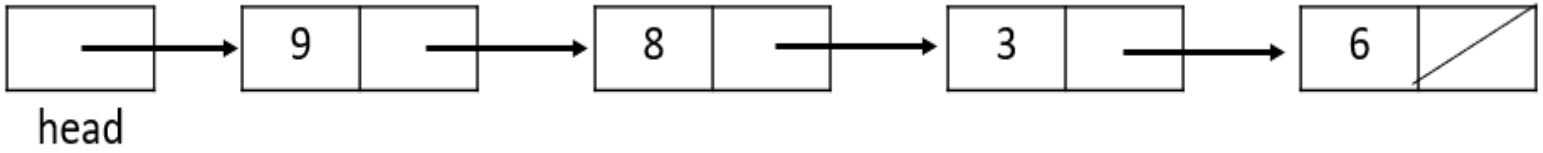


การค้นหาตำแหน่งที่ต้องการ

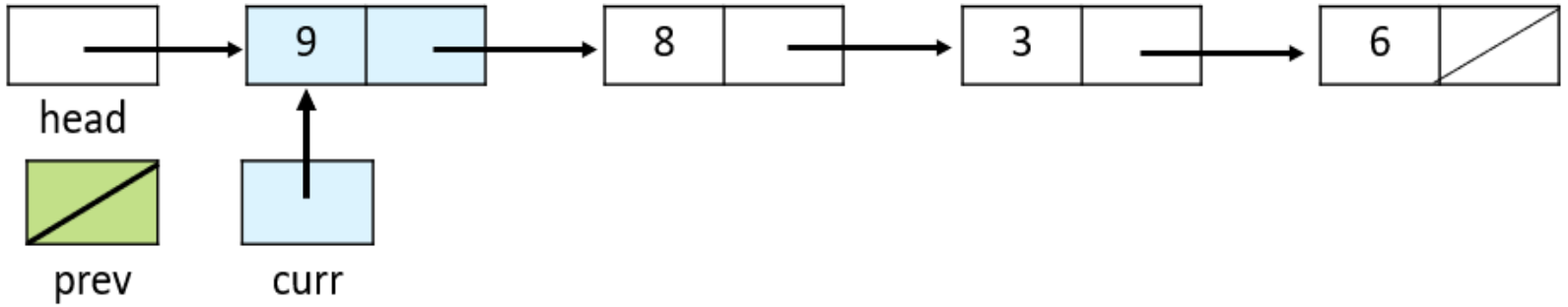
- ก่อนที่จะทำการลบหรือแทรกหนดในลิงค์ลิสต์ จะต้องค้นหาตำแหน่งโหนดที่ต้องการก่อน
- โดยขั้นตอนการค้นหาตำแหน่งโหนดจะมีตัวแปร 2 ตัว เพื่อใช้ในการปรับปรุงโครงสร้างลิงค์ลิสต์ คือ
  1. ตัวแปร curr ทำหน้าที่ค้นหาตำแหน่งโหนดที่ต้องการ
  2. ตัวแปร prev ทำหน้าที่อ้างอิงตำแหน่งโหนดที่อยู่ก่อนโหนดที่ค้นหา
- การค้นหาข้อมูลในลิงค์ลิสต์จะเริ่มทำการค้นหาตั้งแต่ตำแหน่งแรกไปเรื่อย ๆ จนพบตำแหน่งที่ต้องการ หรือถึงสมาชิกตัวสุดท้าย

ข้อมูลที่ค้นหาอยู่ตำแหน่งแรก

- **ต้องการหาตำแหน่งที่เก็บข้อมูล 9** (ข้อมูลดังกล่าวอยู่ตำแหน่งแรก)



- ในการค้นหาตัวแปร curr และตัวแปร prev จะมีสถานะดังภาพ



ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์

- การค้นหาจะเริ่มจากตำแหน่งแรก เมื่อไม่พบก็จะทำการปรับค่าตัวแปร curr และตัวแปร prev
- โดยการอัพเดตค่าตัวแปร prev ให้ทำการเก็บตำแหน่งปัจจุบันไว้ แล้วทำการอัพเดตค่าตัวแปร curr ให้ไปยังโหนดถัดไปด้วยการเรียกเมธอด getLink() ด้วยคำสั่ง

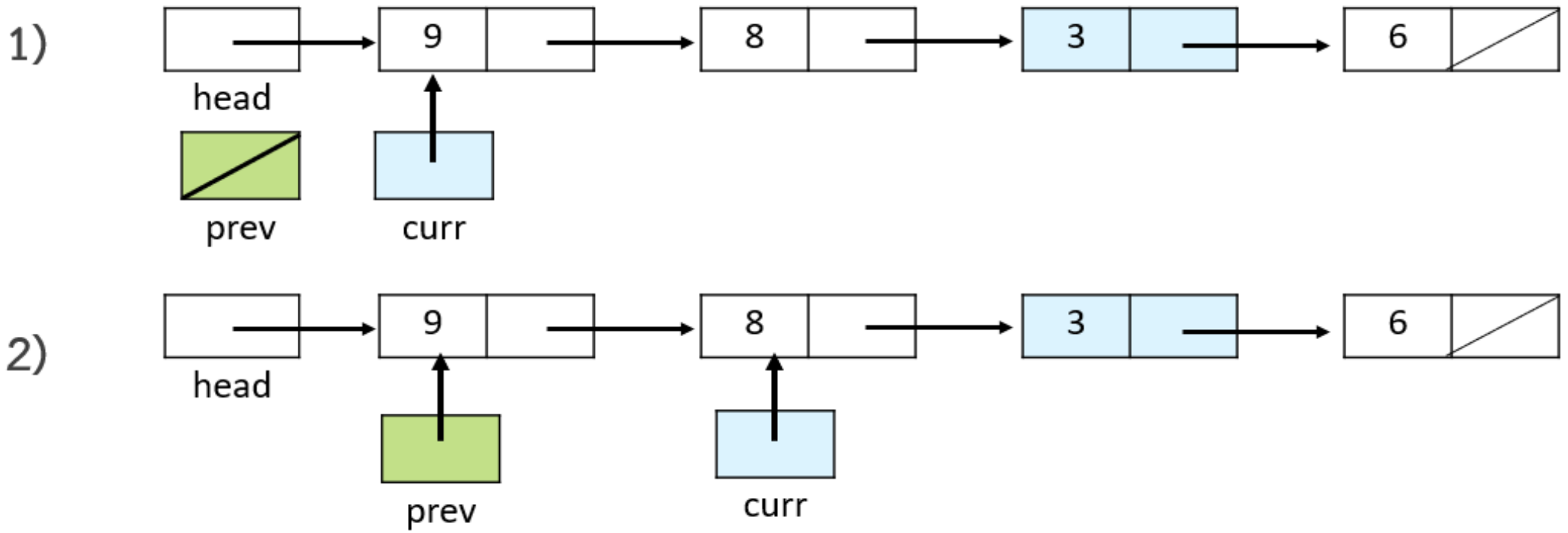
```
prev = curr;

curr = curr.getLink();
```

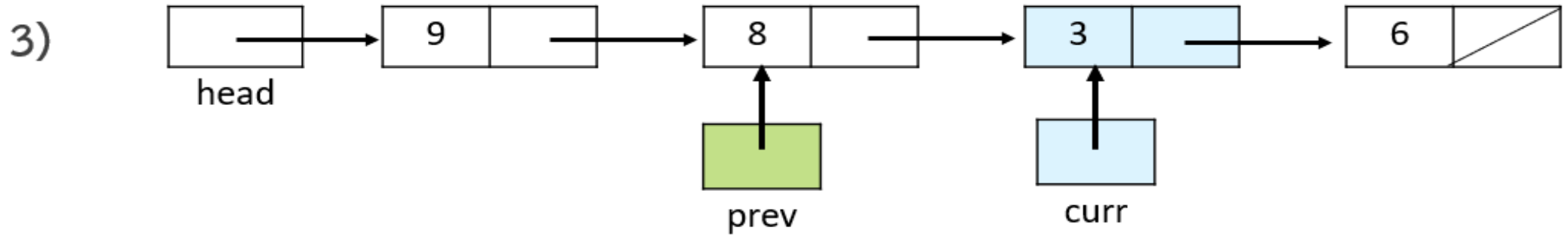
- โดยการทำงานทั้งสองขั้นตอนจะทำควบคู่กันไปจนกระทั่งพบตำแหน่งที่ต้องการ

ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์

ต้องการค้นหาตำแหน่งที่เก็บข้อมูล 3



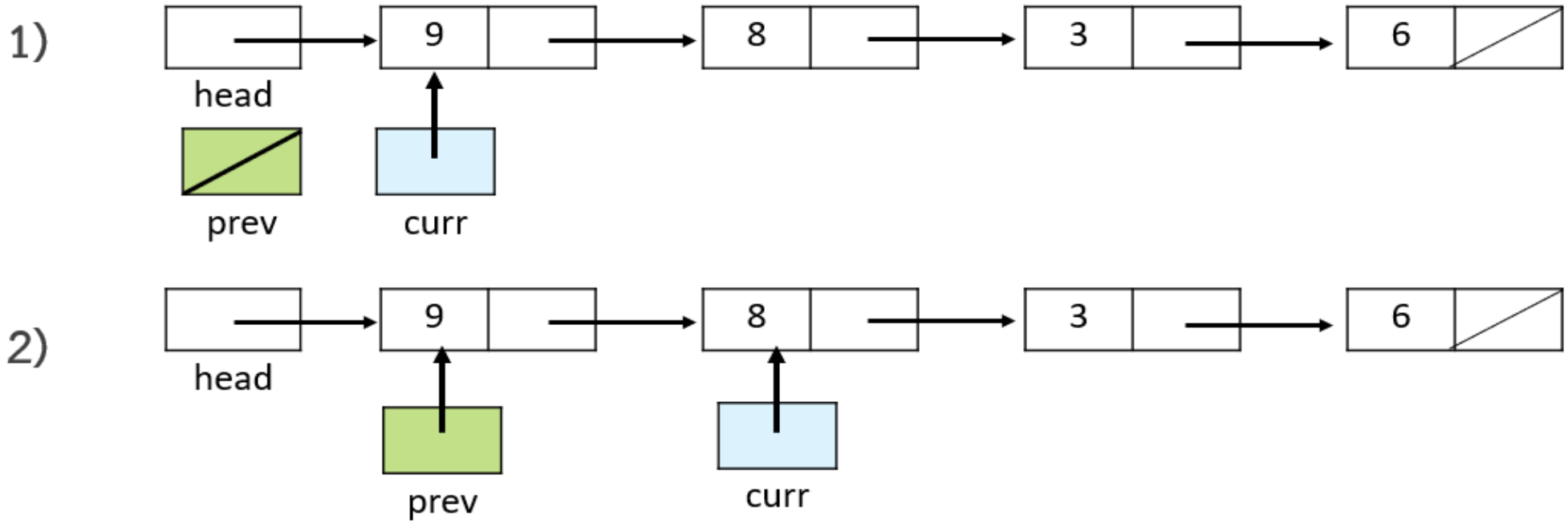
ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์



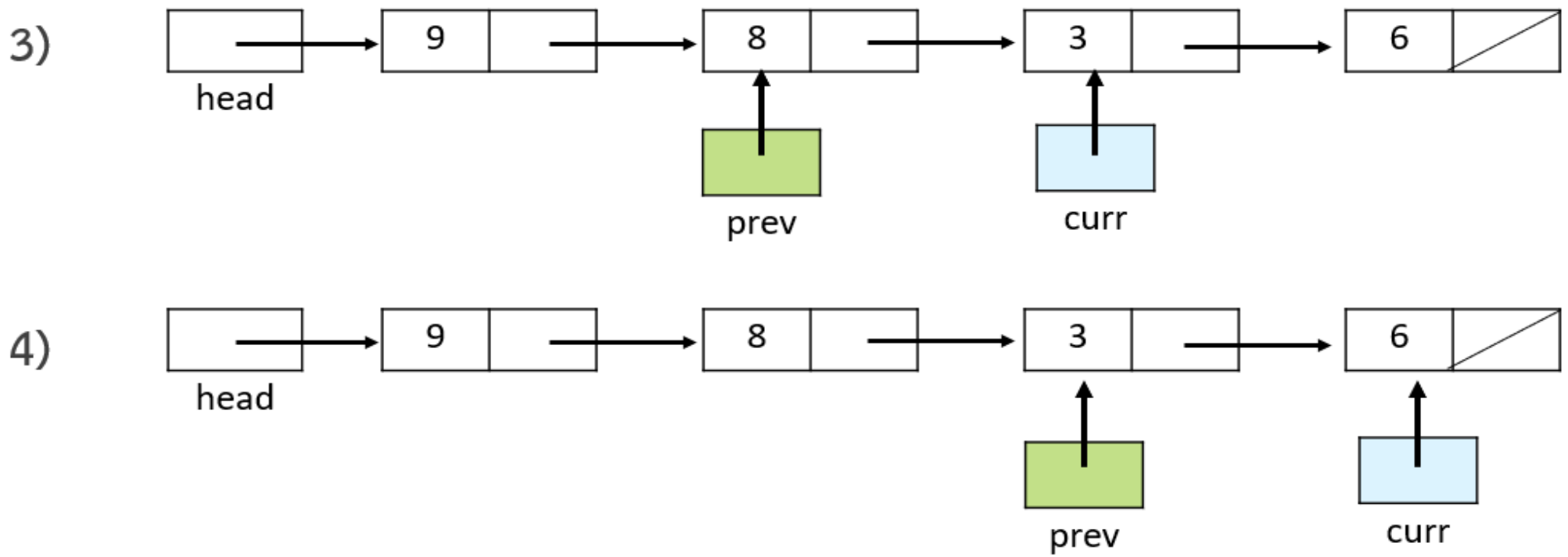
- กรณีที่ไม่พบข้อมูลเมื่อค้นหาจนถึงตำแหน่งสุดท้ายของลิงค์ลิสต์ เมื่อทำการอัพเดตค่าตัวแปรต่อ ตัวแปร curr จะมีค่าเป็น NULL

ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์

ต้องการค้นหาตำแหน่งที่เก็บข้อมูล 7 (กรณีที่ไม่พบข้อมูล)

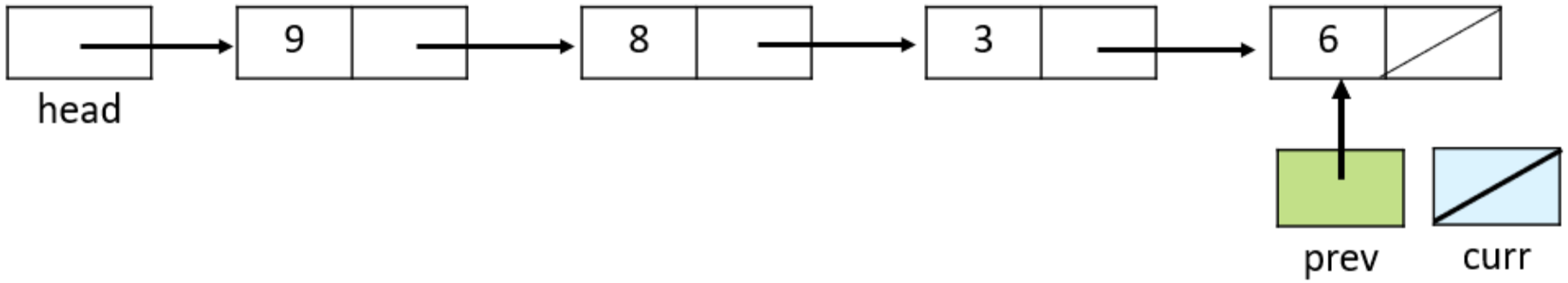


ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์



ข้อมูลที่ค้นหาอยู่ตำแหน่งใดๆ ในลิงค์ลิสต์

5)



การลบโหนด

- การลบโหนดในลิงค์ลิสต์เป็นการยกเลิกการเชื่อมต่อไปยังโหนดที่ต้องการลบ
- เมื่อค้นหาโหนดที่ต้องการลบได้แล้วจะทำการปรับการเชื่อมโยงของโหนดที่เหลือให้ข้ามโหนดที่ลบ
- ขั้นตอนการทำงานสามารถแยกตามตำแหน่งของการลบโหนดในลิงค์ลิสต์ได้เป็น 3 กรณี คือ
  - การลบโหนดที่อยู่ระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์
  - ลบโหนดที่ตำแหน่งแรก
  - ลบโหนดที่ตำแหน่งสุดท้าย

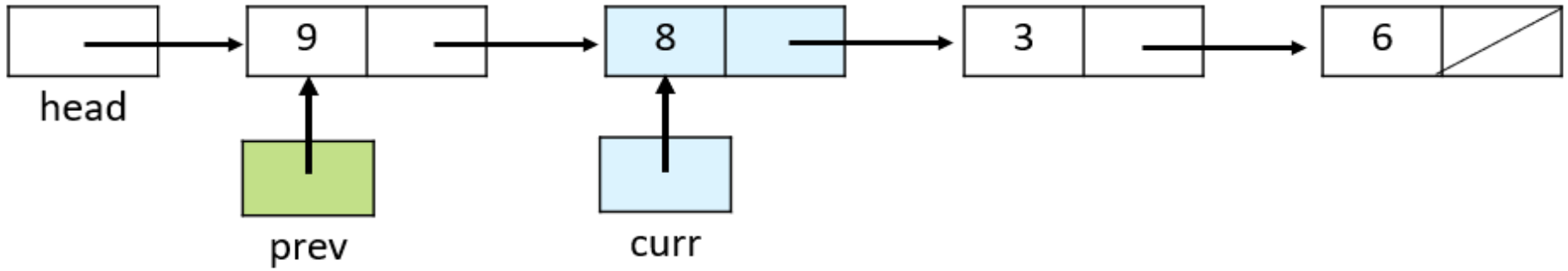
การลบโหนดที่อยู่ระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

- การลบโหนดที่อยู่ระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์ จะต้องค้นหาโหนดที่ต้องการลบก่อน
- โดยตัวแปรที่จำเป็นในการค้นหาตำแหน่งคือตัวแปร curr และ prev
- เมื่อค้นหาข้อมูลโหนดที่ต้องการลบได้แล้ว จะทำการปรับค่าลิงค์ของโหนดที่เหลือ โดยจะทำการปรับค่าลิงค์ของโหนด prev ให้อ้างอิงไปยังโหนดที่อยู่หลังโหนด curr เพื่อลบโหนด curr ออกจากลิงค์ลิสต์ โดยใช้คำสั่ง

```
prev.setLink(curr.getLink());
```

การลบโหนดที่อยู่ระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

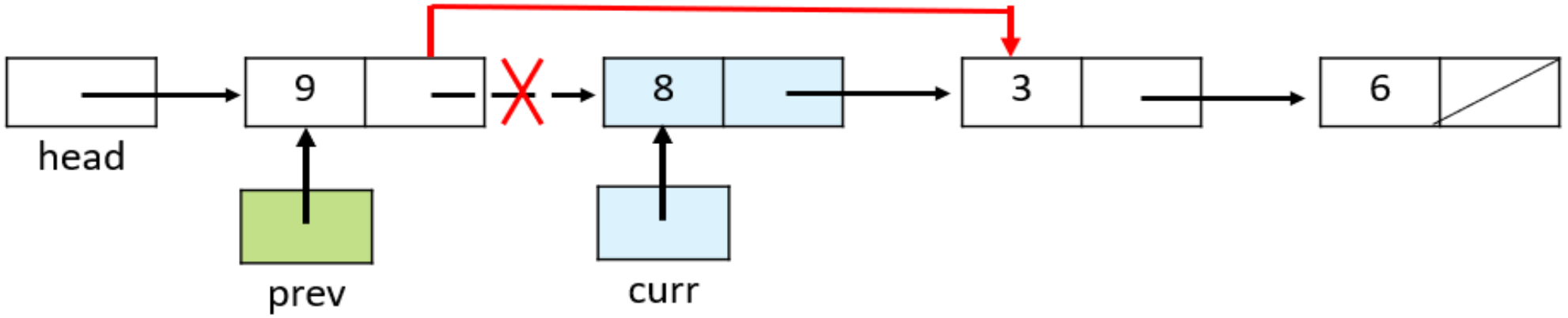
ต้องการลบโหนดที่เก็บข้อมูล 8



- เมื่อค้นหาข้อมูลพบ ตัวแปร curr และ prev จะอยู่ตำแหน่งดังภาพ

การลบโหนดที่อยู่ระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

เมื่อใช้คำสั่ง prev.setLink(curr.getLink()); จะปรากฏเส้นสีแดงขึ้นดังภาพ



การลบโหนดที่อยู่ตำแหน่งแรกของลิงค์ลิสต์

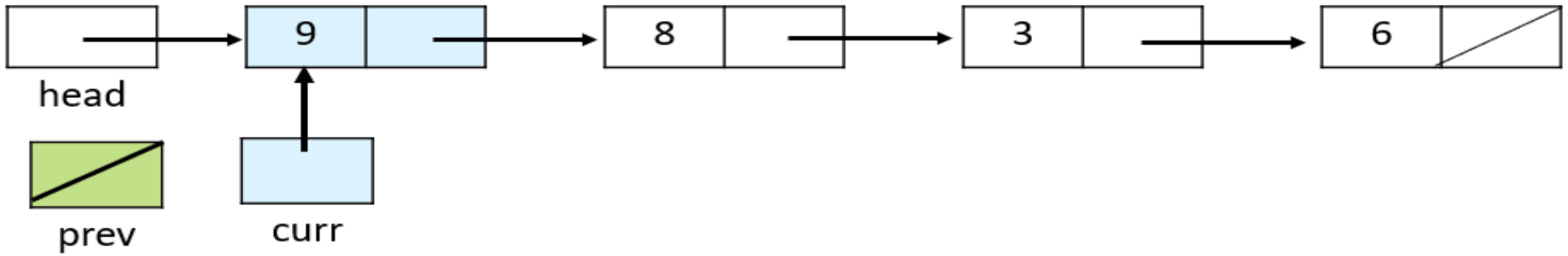
- กรณีที่ทำการค้นหาโหนดที่ต้องการลบแล้วปรากฏว่าโหนดดังกล่าวอยู่ตำแหน่งแรกของลิงค์ลิสต์
- การอัพเดตโครงสร้างลิงค์ลิสต์จะเป็นการอัพเดตค่าลิงค์ของ head เพื่อให้มีค่าเป็นโหนดที่อยู่ถัดไปแทน โดยคำสั่ง

```
head = curr.getLink();
```

การลบโหนดที่อยู่ตำแหน่งแรกของลิงค์ลิสต์

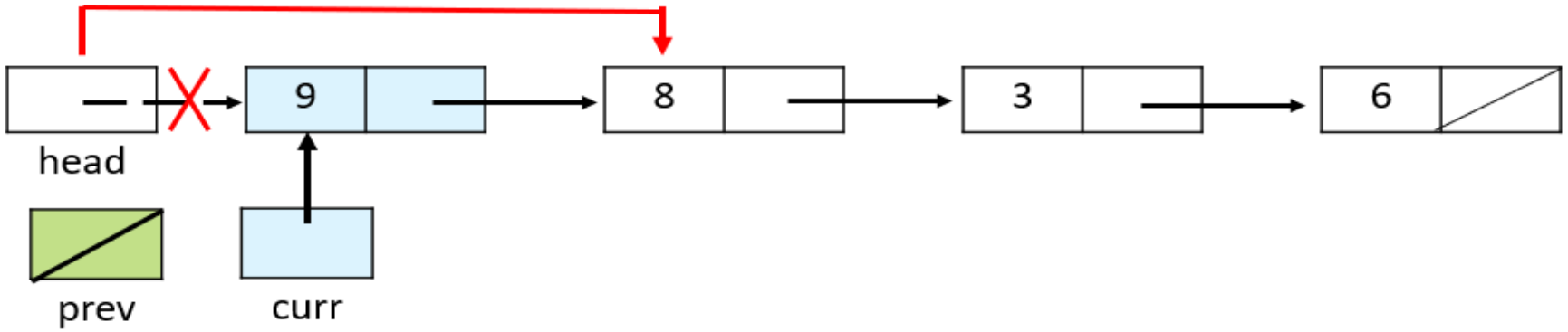
ต้องการลบโหนดที่เก็บข้อมูล 9

- ตัวแปร curr จะใช้ในการอ้างอิงตำแหน่งโหนดที่ต้องการลบและตัวแปร prev จะใช้ในการอ้างอิงตำแหน่งโหนดก่อนหน้าที่ต้องการลบ



การลบโหนดที่อยู่ตำแหน่งแรกของลิงค์ลิสต์

- เมื่อใช้คำสั่ง head = curr.getLink(); จะปรากฏเส้นสีแดงดังภาพ



การลบโหนดที่อยู่ตำแหน่งสุดท้ายของลิงค์ลิสต์

- โหนดที่ต้องการลบอยู่ตำแหน่งสุดท้ายของลิงค์ลิสต์ ในการอัพเดตโครงสร้างลิงค์ลิสต์จะเป็นการอัพเดตค่าลิงค์โหนด prev ซึ่งเป็นโหนดที่อยู่ก่อนหน้าโหนดที่ต้องการลบ

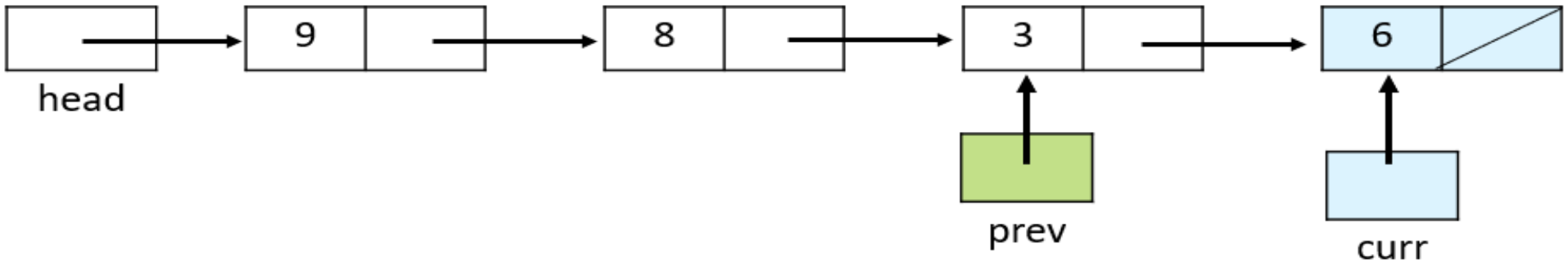
- เพื่อเปลี่ยนโหนด prev ให้เป็นโหนดสุดท้ายแทน โดยใช้คำสั่ง

prev.setLink (null) ; หรือ prev.setLink (curr.getLink()) ;

การลบโหนดที่อยู่ตำแหน่งสุดท้ายของลิงค์ลิสต์

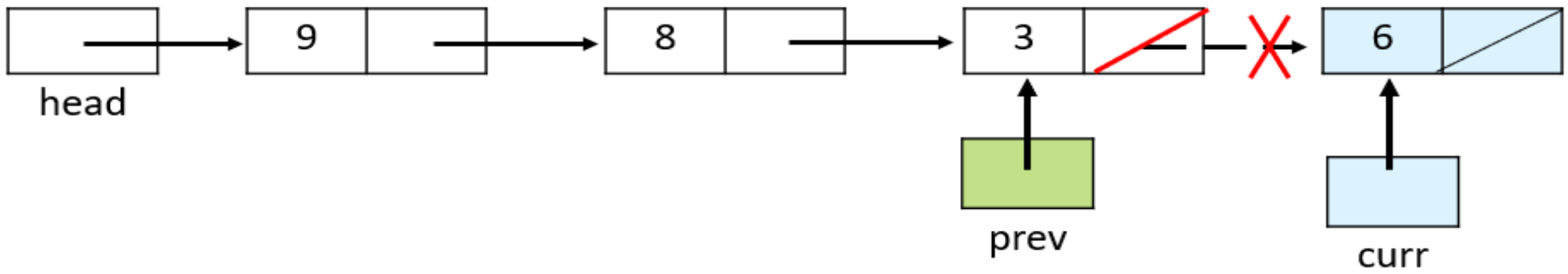
ต้องการลบโหนดที่ทำการเก็บข้อมูล 6

- ซึ่งเมื่อทำการค้นหาแล้วปรากฏว่าอยู่ที่โหนดสุดท้ายของลิงค์ลิสต์ ดังนั้นจึงต้องทำการอัพเดตค่าลิงค์ให้กับโหนด prev เป็น null เพื่อให้โหนด prev กลายเป็นโหนดสุดท้ายแทนดังภาพ



การลบโหนดที่อยู่ตำแหน่งสุดท้ายของลิงค์ลิสต์

- เมื่อใช้คำสั่ง prev.setLink (null) ; จะปรากฏเส้นสีแดงดังภาพ



การแทรกโหนด

- เมื่อค้นหาตำแหน่งโหนดที่ต้องการได้แล้วจะทำการสร้างเส้นเชื่อมใหม่ระหว่างโหนดใหม่กับโครงสร้างเดิม โดยการอัพเดตค่าลิงค์ของโหนดที่เกี่ยวข้อง
- ขั้นตอนการทำงานสามารถแยกตามตำแหน่งของการแทรกโหนดในลิงค์ลิสต์ได้เป็น 3 กรณี คือ
  - การแทรกโหนดเข้าระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์
  - การแทรกโหนดเข้าที่ตำแหน่งแรกของลิงค์ลิสต์
  - การเพิ่มโหนดในตำแหน่งสุดท้ายของลิงค์ลิสต์

การแทรกโหนดเข้าระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

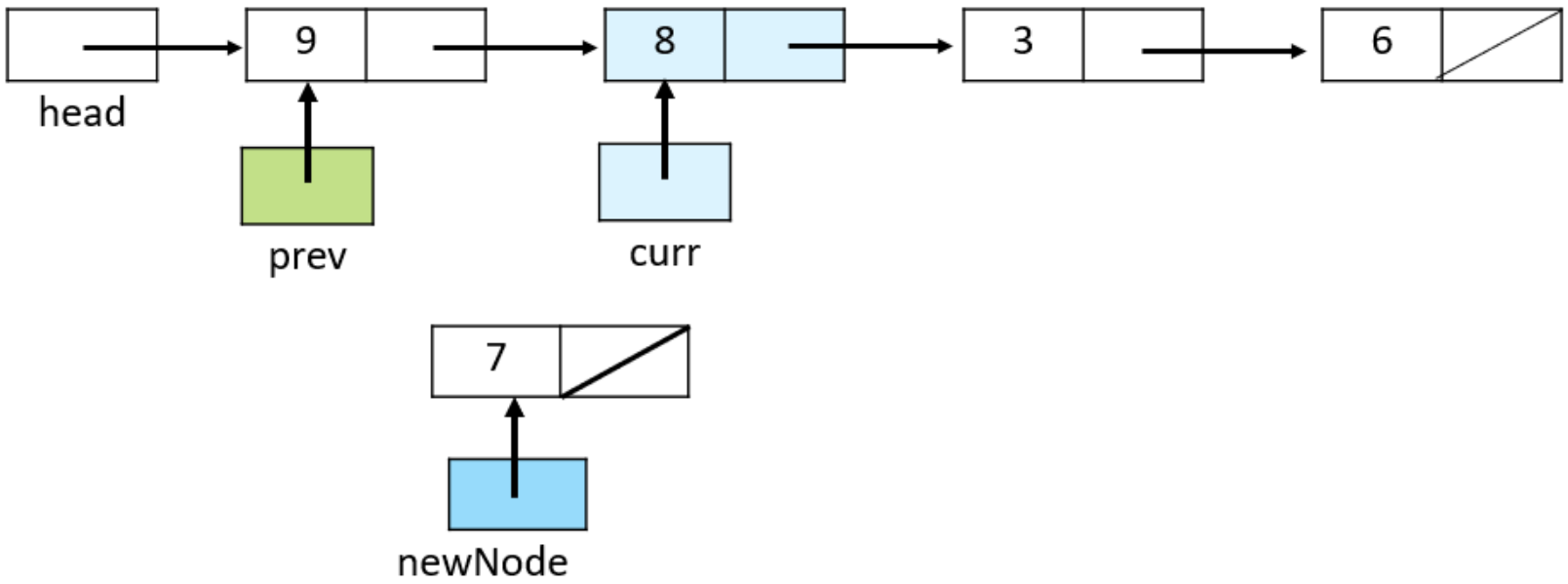
- เมื่อค้นหาโหนดที่ต้องการได้แล้วจะทำการเชื่อม newNode เข้ากับลิงค์ลิสต์
- โดยกำหนดค่าลิงค์ของ newNode ให้มีค่าเป็น curr และกำหนดค่าลิงค์ของโหนด prev ให้มีค่าเป็น newNode ด้วยคำสั่ง

```
newNode.setLink(curr);

prev.setLink(newNode);
```

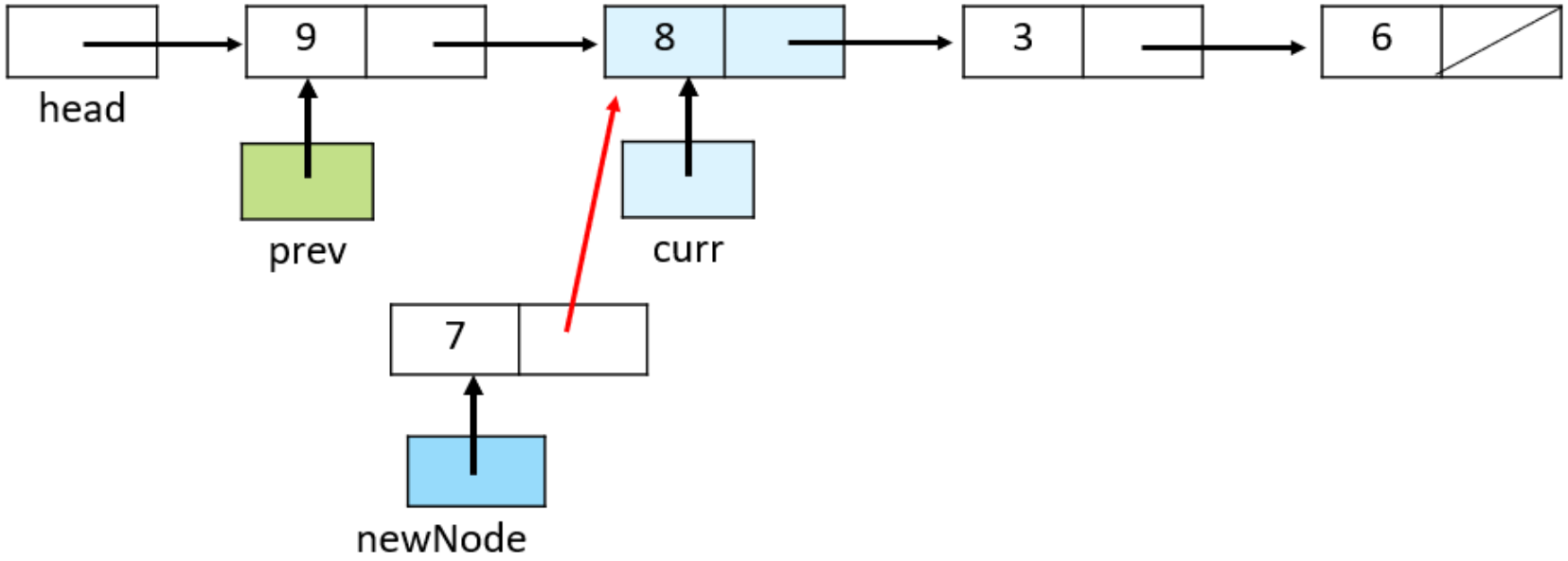
การแทรกโหนดเข้าระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

ต้องการแทรกโหนด newNode ที่มีค่าข้อมูล 7 เข้าระหว่างโหนด curr และ prev



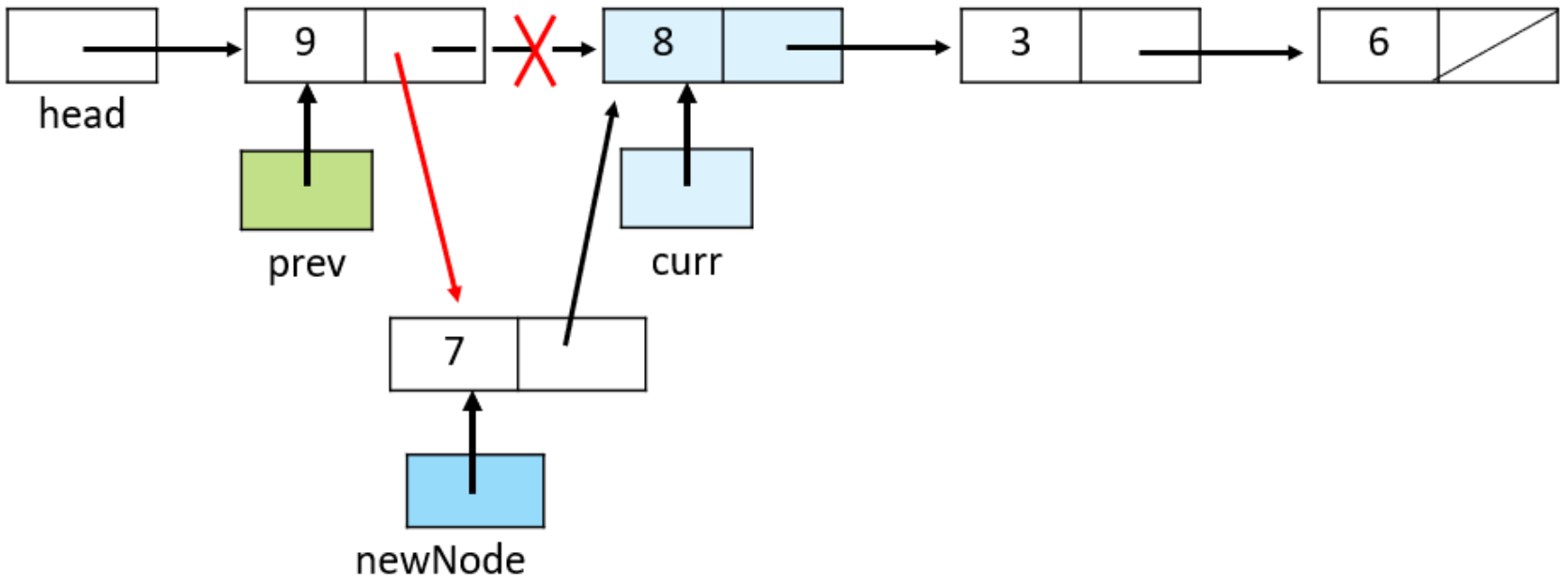
การแทรกโหนดเข้าระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

- เมื่อใช้คำสั่ง newNode.setLink(curr); จะปรากฏเส้นสีแดงดังภาพ



การแทรกโหนดเข้าระหว่างข้อมูลที่มีอยู่ในลิงค์ลิสต์

- เมื่อใช้คำสั่ง prev.setLink(newNode); จะปรากฏเส้นสีแดงดังภาพ



การแทรกโหนดเข้าที่ตำแหน่งแรกของลิงค์ลิสต์

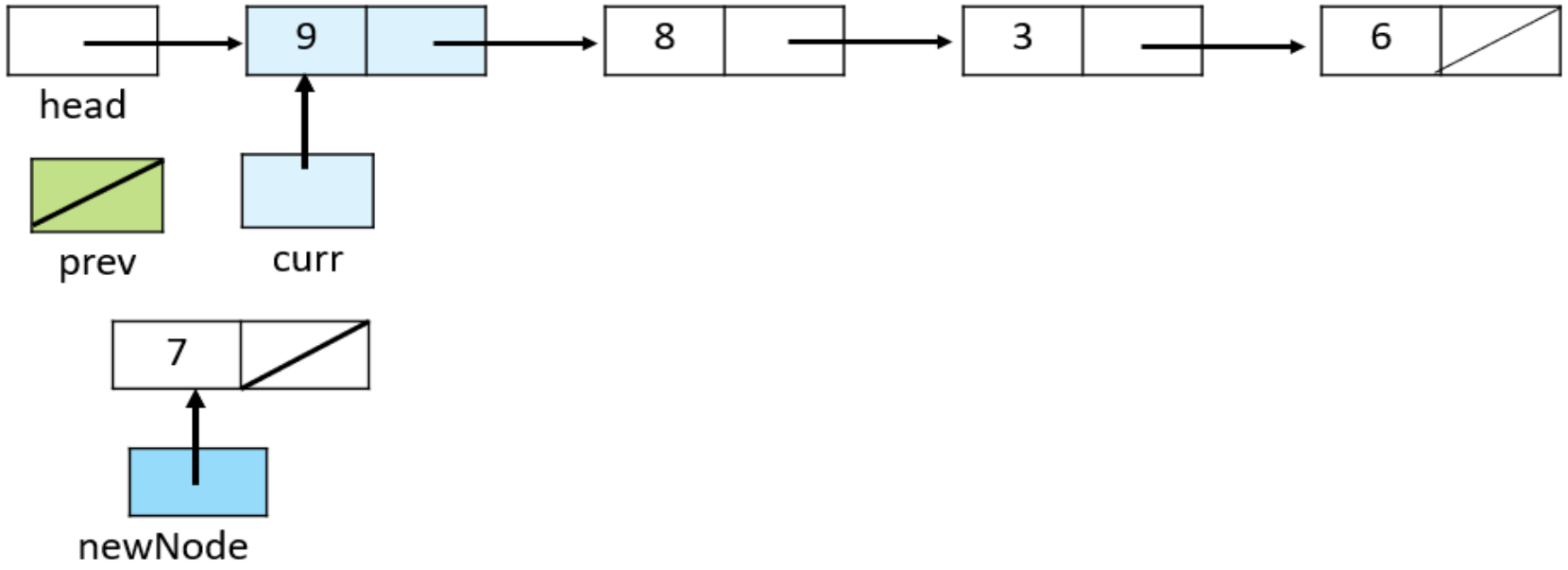
- กรณีที่ต้องการแทรกโหนดเข้าไปที่ตำแหน่งแรกของลิงค์ลิสต์
- ในการอัพเดตโครงสร้างลิงค์ลิสต์นอกจากกำหนดค่าลิงค์ให้กับโหนด newNode เป็น curr แล้วยังต้องทำการอัพเดตค่าลิงค์ของ head เพื่อให้มีค่าเป็นโหนด newNode แทน โดยใช้คำสั่ง

```
newNode.setLink(curr);

head.setLink(newNode);
```

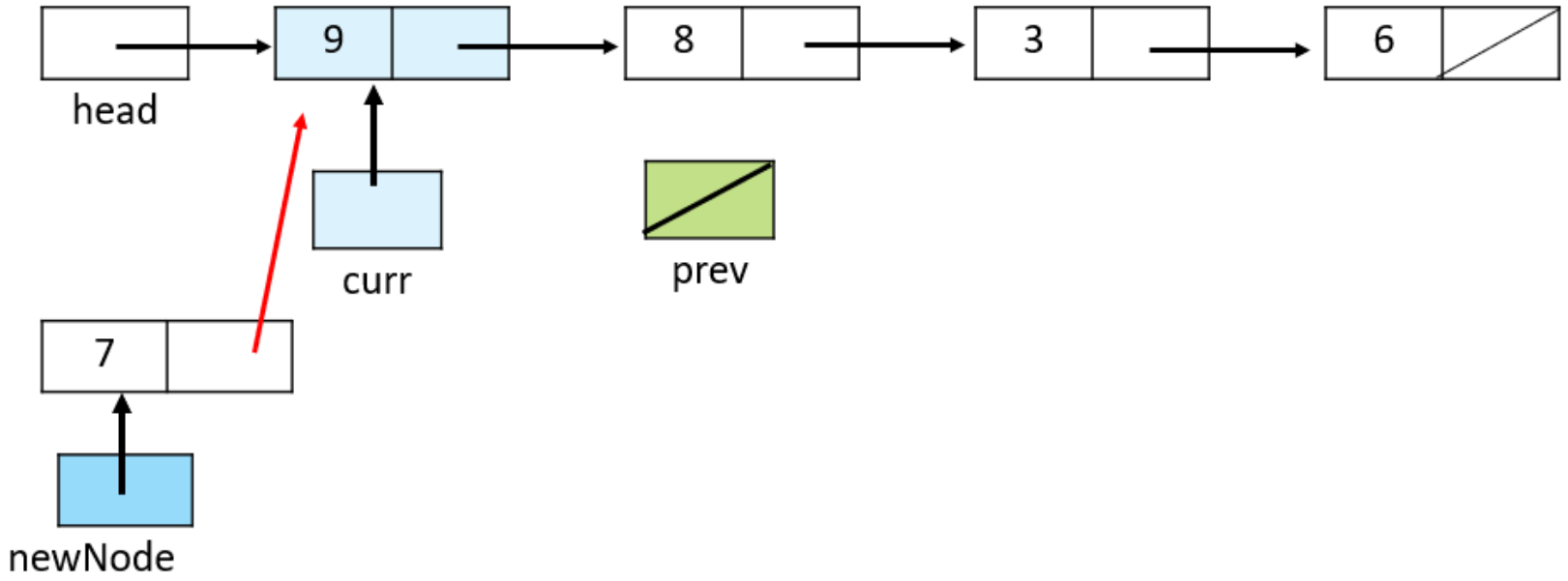
การแทรกโหนดเข้าที่ตำแหน่งแรกของลิงค์ลิสต์

- โดยตัวแปร curr จะใช้ในการอ้างอิงตำแหน่งโหนดทางขวาของโหนดที่จะเพิ่มและตัวแปร prev จะใช้อ้างอิงตำแหน่งโหนดซ้ายของโหนดที่จะแทรก



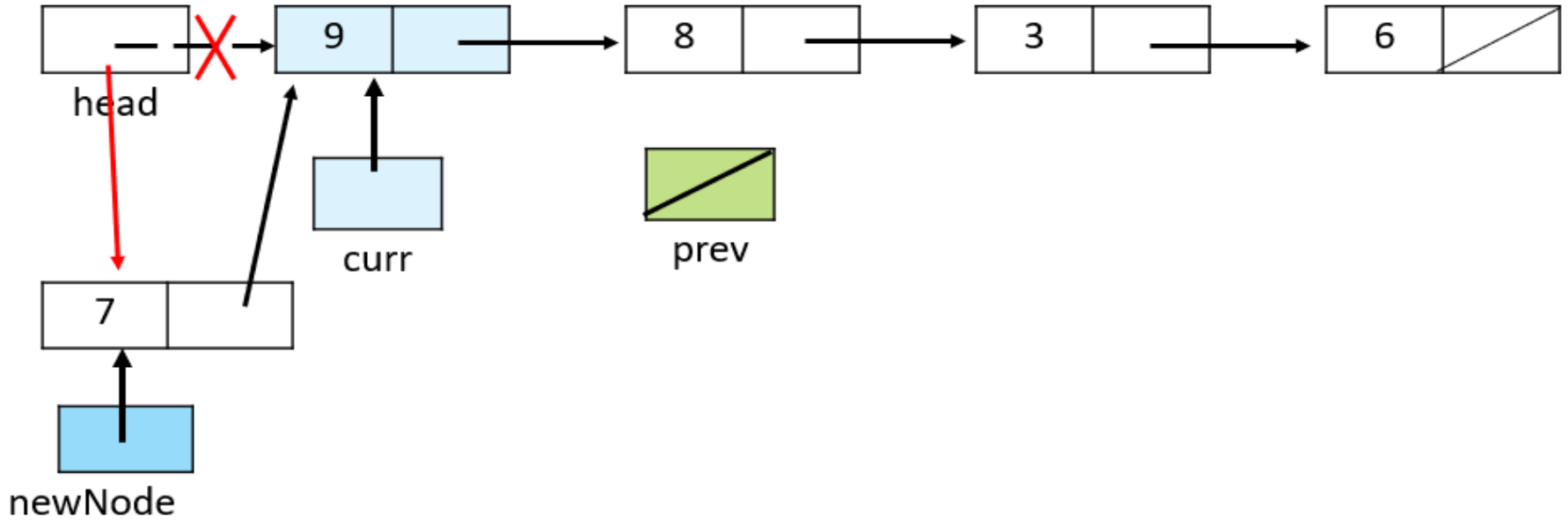
การแทรกโหนดเข้าที่ตำแหน่งแรกของลิงค์ลิสต์

- เมื่อใช้คำสั่ง newNode.setLink(curr); จะปรากฏเส้นสีแดงดังภาพ



การแทรกโหนดเข้าที่ตำแหน่งแรกของลิงค์ลิสต์

- เมื่อใช้คำสั่ง head.setLink(newNode); จะปรากฏเส้นสีแดงดังภาพ



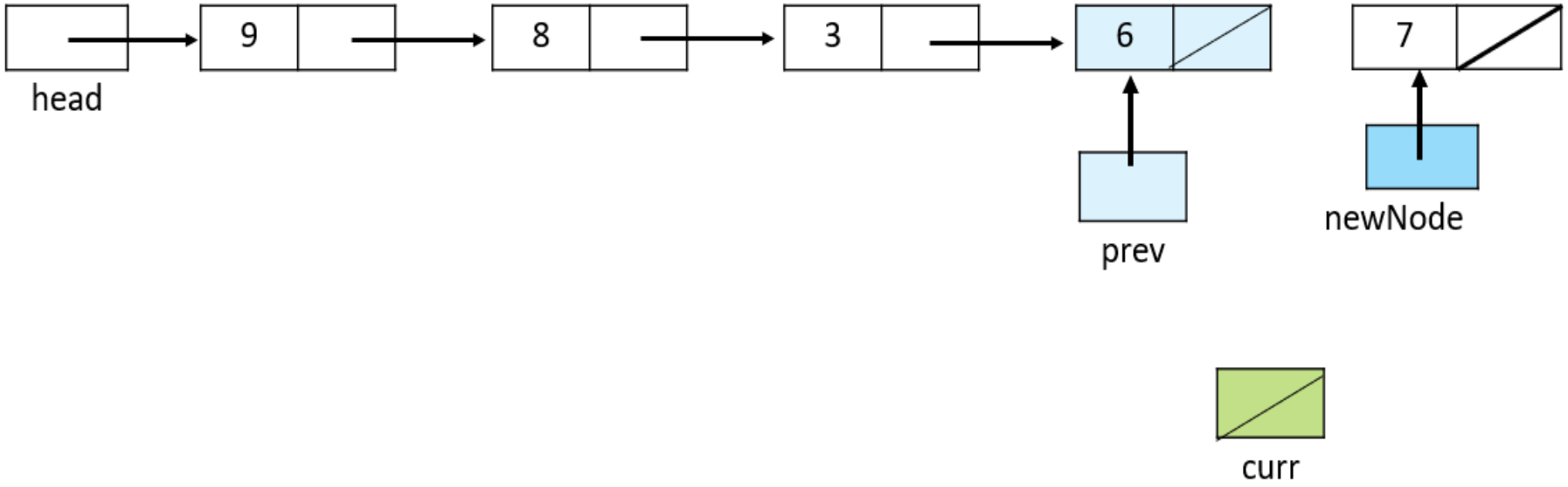
การเพิ่มโหนดเข้าที่ตำแหน่งสุดท้ายของลิงค์ลิสต์

- การเพิ่มโหนดในตำแหน่งสุดท้ายของลิงค์ลิสต์จะต้องทำการค้นหาโหนดสุดท้ายก่อน
- โดยจะใช้ตัวแปรช่วย 2 ตัวในการค้นหาตำแหน่งคือตัวแปร curr และ prev เพื่อใช้ในการเดินทางบนลิงค์ลิสต์เพื่อหาตำแหน่งโหนดสุดท้าย
- เมื่อพบตำแหน่งที่ต้องการตัวแปร prev จะทำหน้าที่ระบุตำแหน่งของโหนดสุดท้าย ดังนั้นการจะเพิ่มโหนดใหม่เข้าไปจะทำการอัพเดตค่าลิงค์ของโหนด prev ให้เชื่อมไปยังโหนด newNode โดยใช้คำสั่ง

```
prev.setLink(newNode);
```

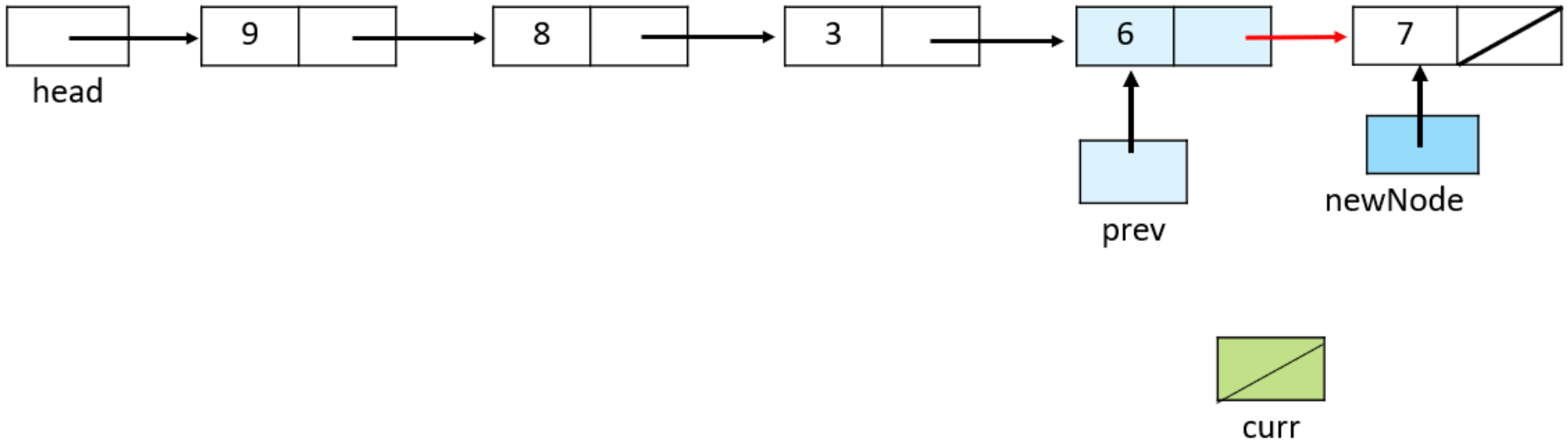
การเพิ่มโหนดเข้าที่ตำแหน่งสุดท้ายของลิงค์ลิสต์

- เมื่อเดินทางไปถึงตำแหน่งสุดท้ายของลิงค์ลิสต์ ตัวแปร prev และ curr จะมีสถานะดังภาพ



การเพิ่มโหนดเข้าที่ตำแหน่งสุดท้ายของลิงค์ลิสต์

- เมื่อใช้คำสั่ง prev.setLink(newNode); จะปรากฏเส้นสีแดงดังภาพ



การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

- การใช้ลิงค์ลิสต์ทางเดียวเก็บข้อมูลของสแต็กจะแตกต่างจากอาเรย์ที่ไม่ต้องมีการตรวจสอบว่ามีพื้นที่สำหรับรับข้อมูลเพิ่มได้หรือไม่
- เนื่องจากลิงค์ลิสต์ไม่มีข้อจำกัดเรื่องขนาด
- เมื่อมีสมาชิกใหม่เข้ามาก็จะต้องทำการสร้างโหนดใหม่สำหรับเก็บข้อมูลที่จะเพิ่มลงไป

การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

- การพุชจะมีพารามิเตอร์ที่ต้องส่งมา 1 ตัว คือ ข้อมูลที่ต้องการเก็บลงสแต็ก
- ในการทำงานจะมีตัวแปรที่เกี่ยวข้องดังนี้
  - item จะเป็นข้อมูลที่ต้องการนำใส่สแต็ก
  - head ทำหน้าที่ระบุตำแหน่งแรกของลิงค์ลิสต์ที่ใช้เก็บข้อมูลของสแต็ก นั่นคือข้อมูลตัวบนสุดของสแต็กนั่นเอง

การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

PUSH(item):

1. สร้างโหนดใหม่ (newNode) เพื่อเก็บข้อมูล item
2. ตรวจสอบว่าในลิงค์ลิสต์มีข้อมูลหรือไม่ โดยตรวจสอบว่า head เป็น null หรือไม่
3. ถ้าไม่มีข้อมูลในลิงค์ลิสต์ ให้กำหนด head = newNode

การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

4. ถ้ามีข้อมูลในลิงค์ลิสต์ ให้ทำดังนี้

    4.1 กำหนดค่าลิงค์ของ newNode เป็น head

    4.2 กำหนดค่า head = newNode

การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

- การป็อปจะไม่มีการส่งค่าพารามิเตอร์ใด ๆ มา เนื่องจากเป็นการนำข้อมูลตัวสุดท้ายของสแต็กออกเสมอ
- ตัวแปรที่เกี่ยวข้องดังนี้
  - item ทำหน้าที่เก็บโหนดที่นำออกจากสแต็ก
  - head ทำหน้าที่ระบุตำแหน่งแรกของลิงค์ลิสต์ที่ใช้เก็บข้อมูลของสแต็ก

การทำงานของ stack บนโครงสร้างลิงค์ลิสต์ทางเดียว

POP() :

1. ตรวจสอบว่าในลิงค์ลิสต์มีข้อมูลหรือไม่ โดยตรวจสอบว่า head เป็น null หรือไม่
2. ถ้ามีให้คัดลอกโหนดที่ตำแหน่งแรกไว้ : item = head.getItem();
3. ปรับค่า head ให้ชี้โหนดถัดไป : head = head.getLink();
4. ส่งค่าข้อมูลของโหนดที่คัดลอกไว้กลับไป : return item.getItem();