DOUBLE LINKED LIST

Teaching Team of Algorithm and Data Structure

Learning Outcome

- Students must be able to understand basic concept of double linked list

- Students must be able to have a good knowledge of steps for implementing double linked list to solve a problem

Topics

- Double linked list basic concept
- Declaring double linked list
- Adding node
- Deleting node
- Inserting node

Double Linked List

- Double : means there are 2 fields of pointer(link). One points to the next node while another points to the previous node.
- Traversal can run in 2 ways (start form head to tail, or start form tail to head)
- Pointer **next** : links current node to the next node.
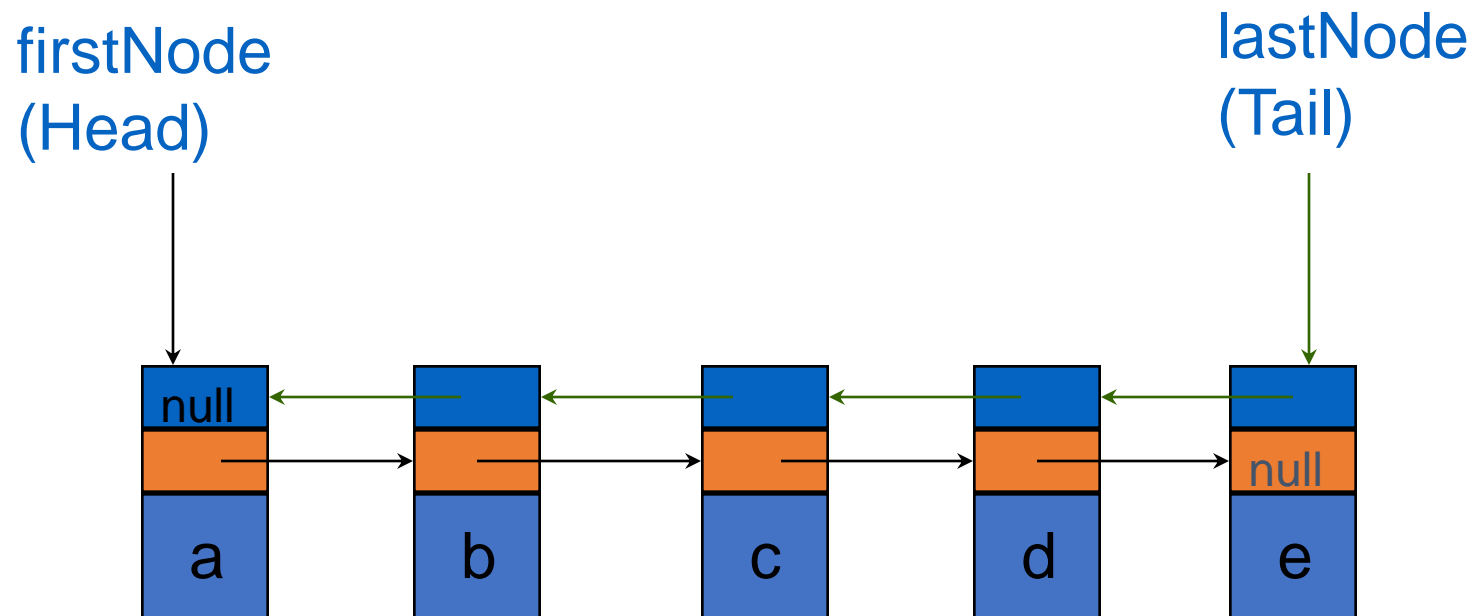- Pointer **prev** : links current node to the previous node.

Node "Double"

prev	DATA	next

- If there is only one node, then next and prev will point to NULL

- Feasible to perform 2 ways traversal:
  1. head to tail
  2. tail to head

Double Linked List

- First node (head) has the pointer prev that points to NULL.
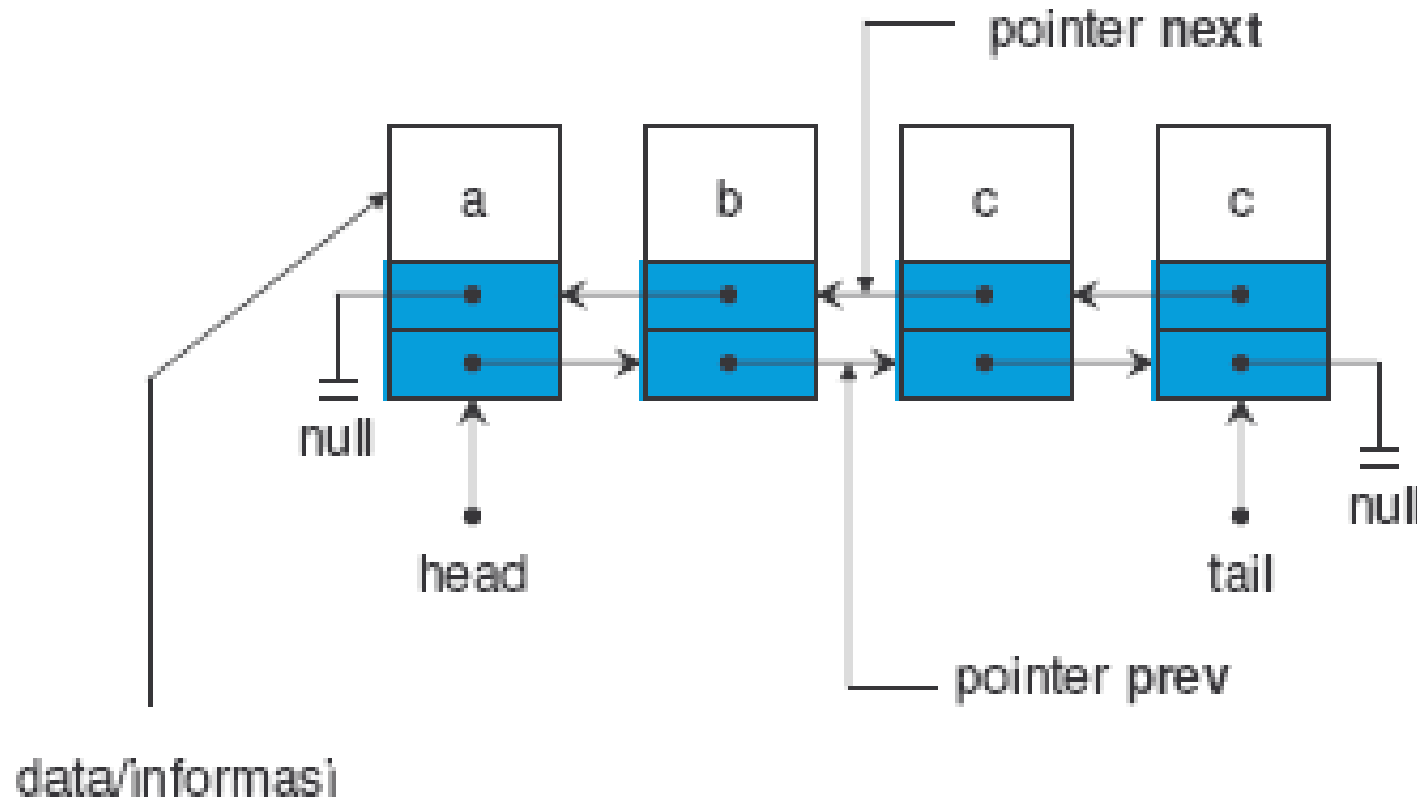- Last node (tail) has the pointer next that points to NULL.

Head and Tail

- Head : a reference to the first node
- Tail : a reference to the last node
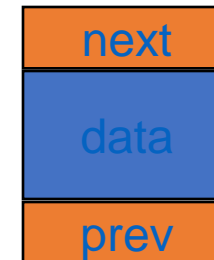
Double Linked List

- Example: There are 4 nodes in a double linked list:

"Double" Representation

```
class Node2P
{
        int data; // data
        Node2P next;        // pointer next
        Node2P prev;        // pointer prev
}
```

Illustration :

- **data** : save a data.
- **next**: refers/links to the next node
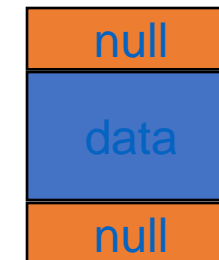- **prev**: refers/links to the previous node

Class Node

```
public class Node2P {
    int data;
    Node2P next;
    Node2P previous;

    Node2P() {}

    Node2P(int theData)
        { data = theData;}

    Node2P(int theData, Node2P thePrevious, Node2P theNext)
    {
        data= theData;
        prev = thePrevious;
        next = theNext;
    }
}
```
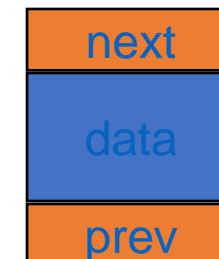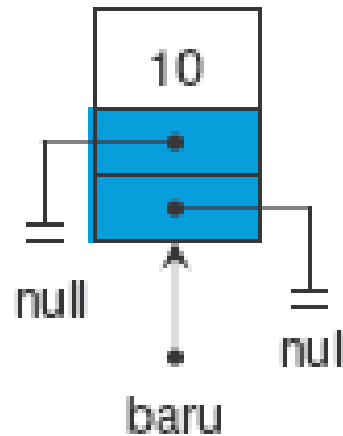
Constructor 1

null
null
null

Constructor 2

null
data
null

Constructor 3

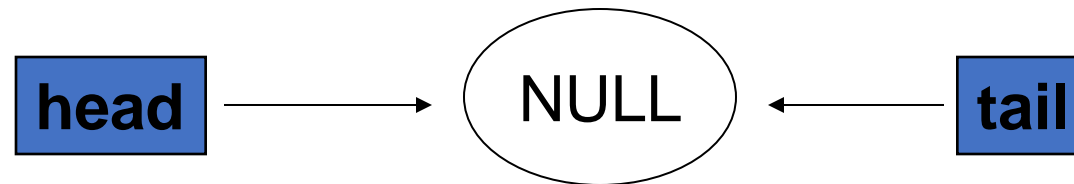| next |
|------|
| data |
| prev |
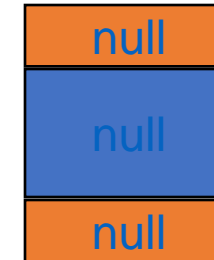
Node Instantiation

Node2P baru = new Node2P(10);

Linked List Operation

1. initialization
2. isEmpty
3. size
4. addition
5. deletion
6. insertion
7. sarching
8. accessing

(1) Initialization

Node2P head = null;
Node2P tail= null;

There is still no node in a double linked list

head → NULL ← tail

ALGORITMA DAN STRUKTUR DATA

(2)isEmpty

Is a double linked list currently empty?

```
boolean isEmpty()
{
    return size==0;
}
```

ALGORITMA DAN STRUKTUR DATA

(3) size

This method will return the size of double linked list. Or how many node is in a linked list?

```
int size()
{
    return size;
}
```

(4) Addition

1. Add First
2. Add Last
3. Add After/Before a node

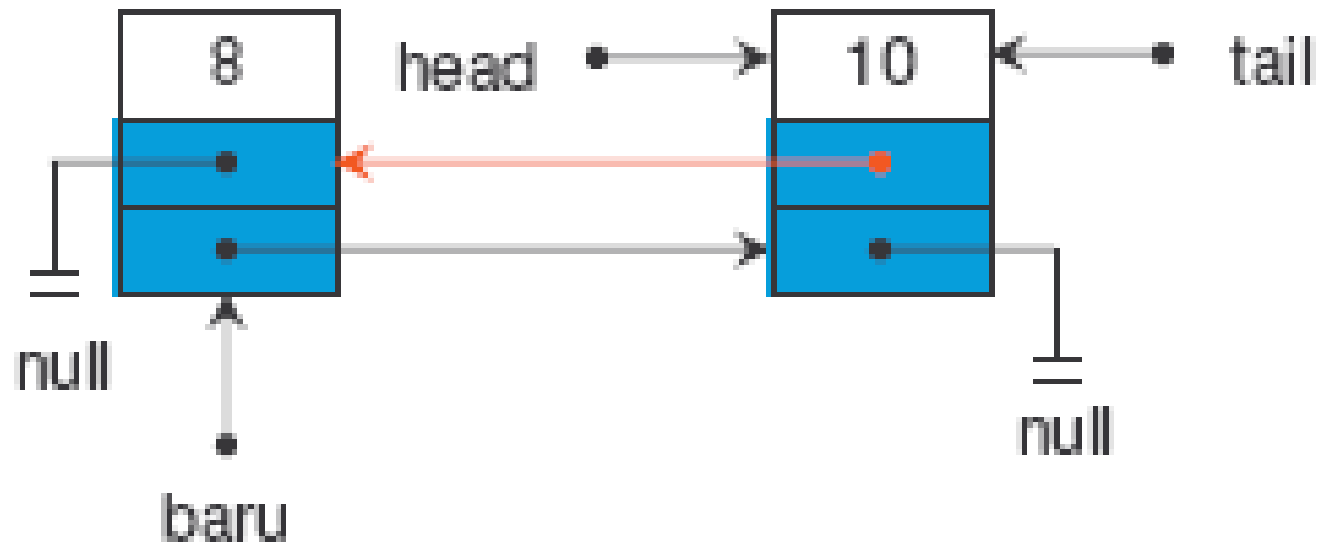# Add a node in an empty linked list

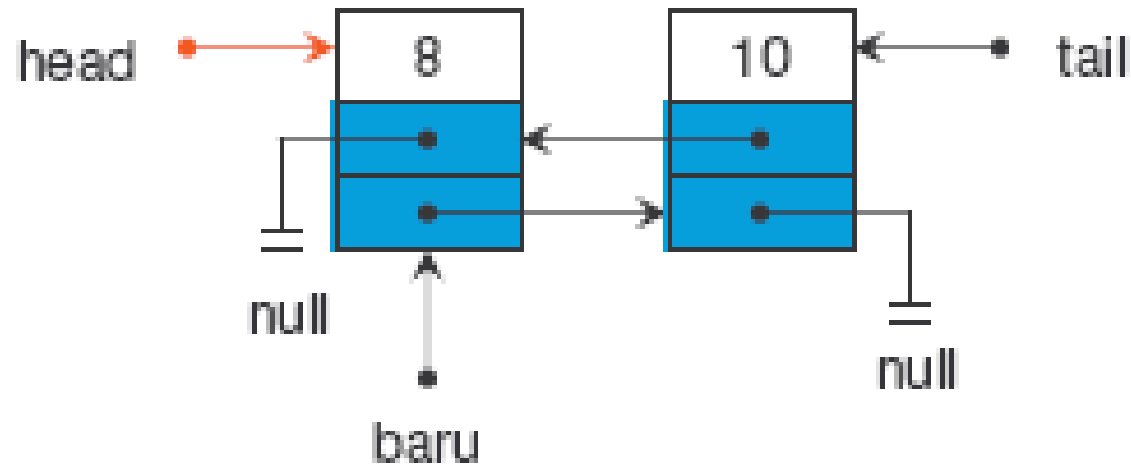- head = tail= baru
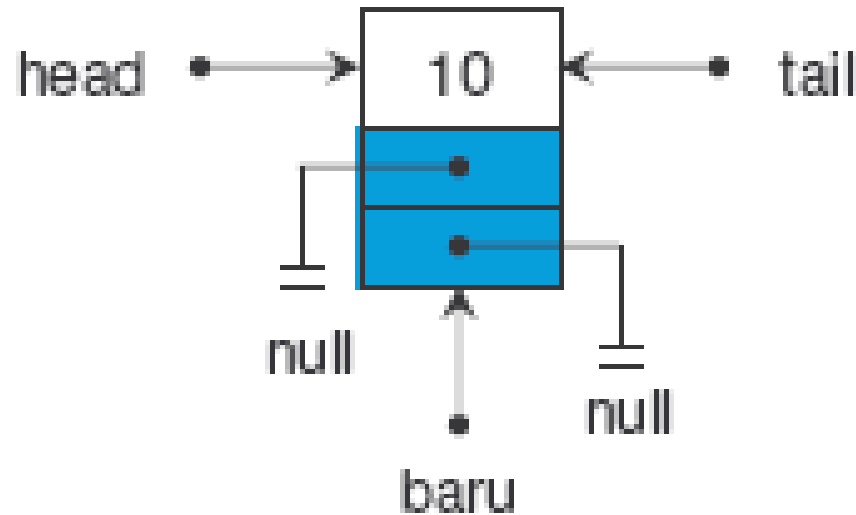
Add First

1. Baru.next = head

ALGORITMA DAN STRUKTUR DATA
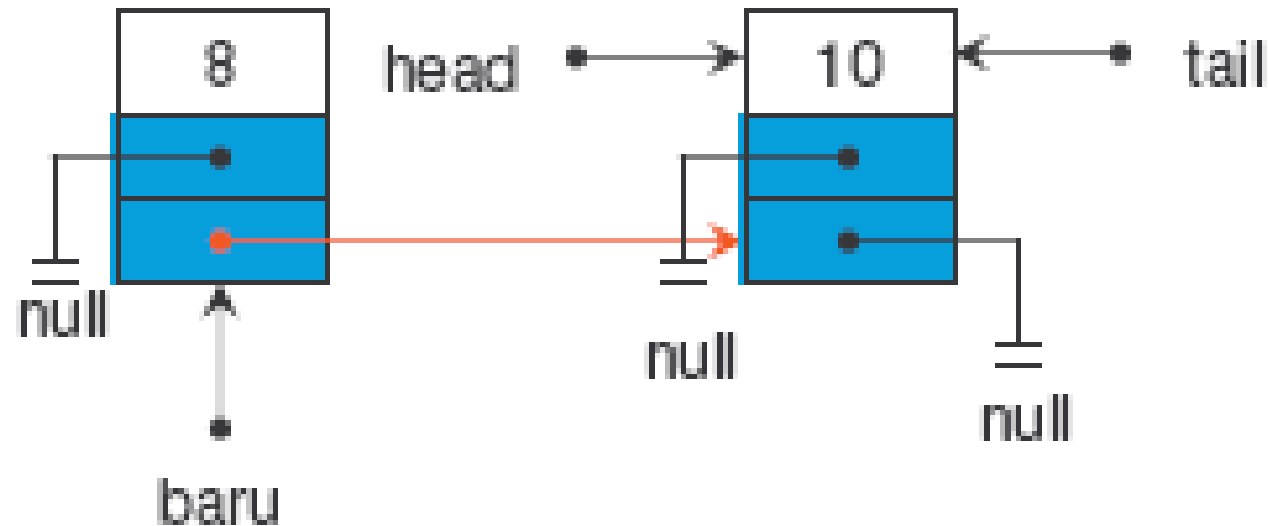
Add First

2. head.prev = Baru

Add First

3. head = baru

Add First
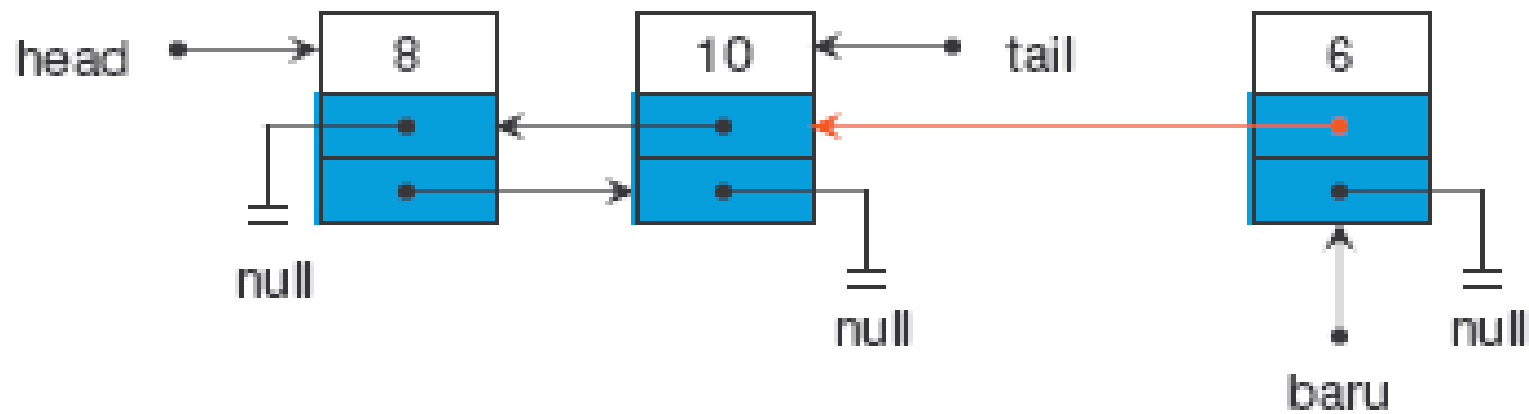
```
void addFirst(Node2P input){
    if (isEmpty()){
        head=input;
        tail=input;
    }
    else
    {
        input.next = head;
        head.previous = input;
        head = input;
    } size++;
}
```
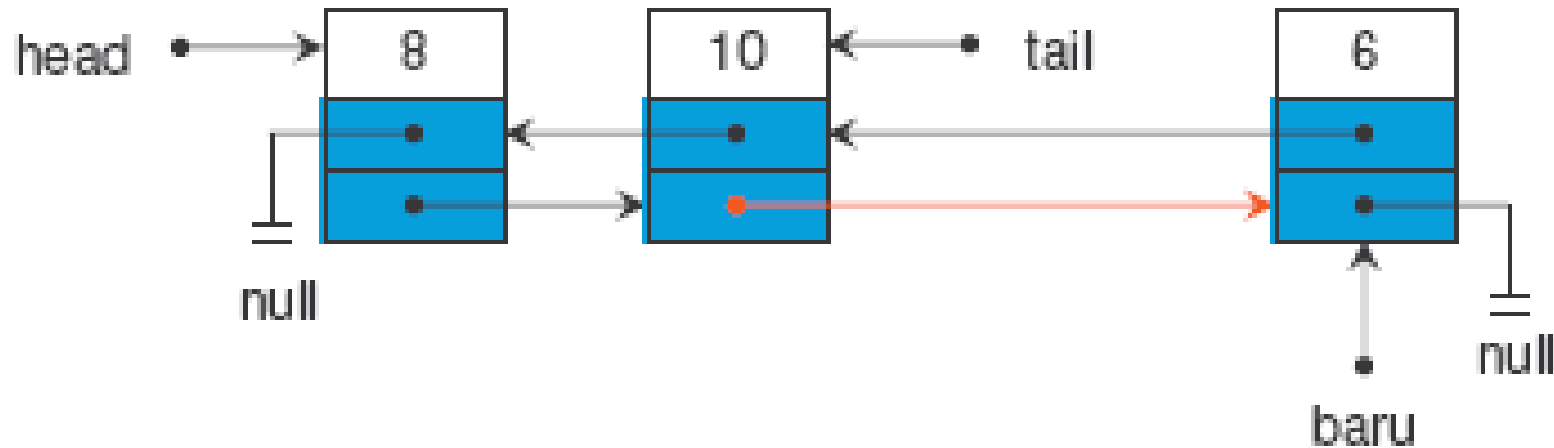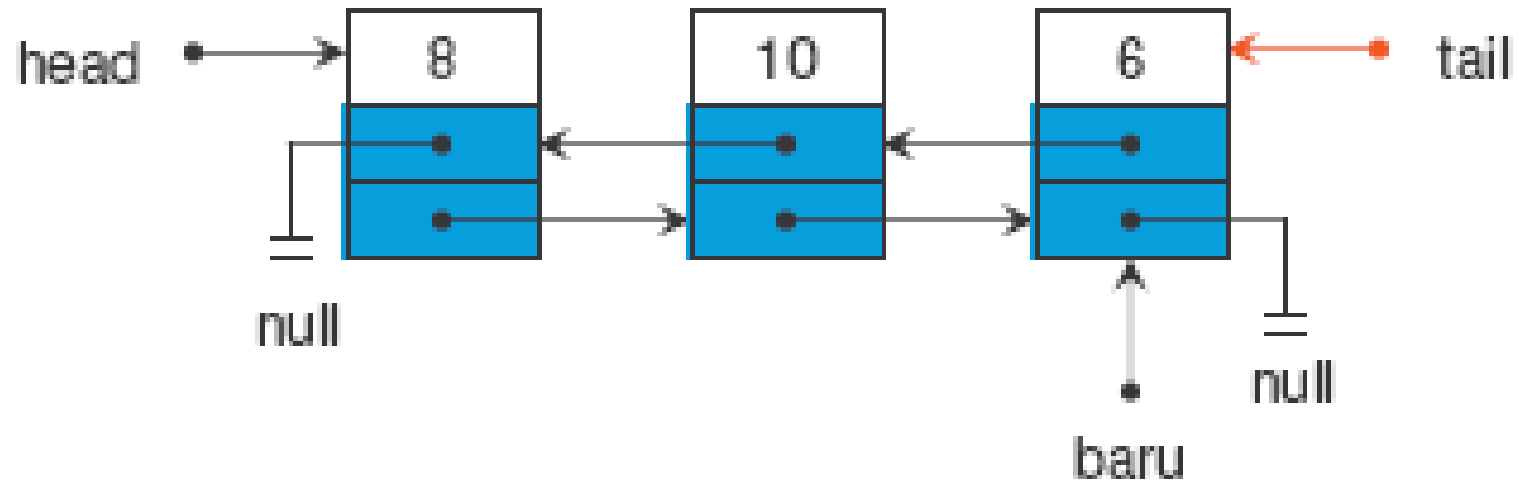
ALGORITMA DAN STRUKTUR DATA
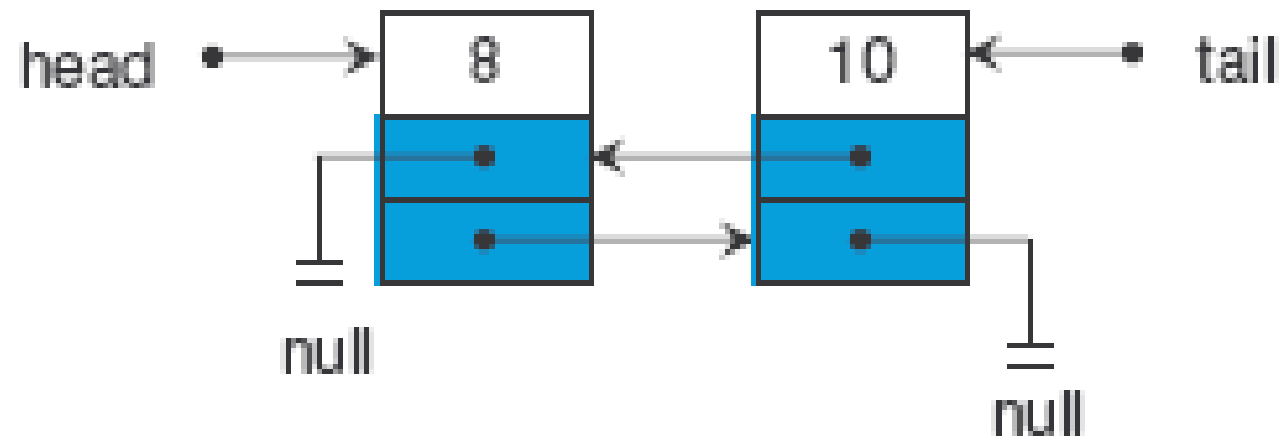
Add Last

Add Last

1. Baru.prev = tail

Add Last

2. tail.next = Baru

ALGORITMA DAN STRUKTUR DATA
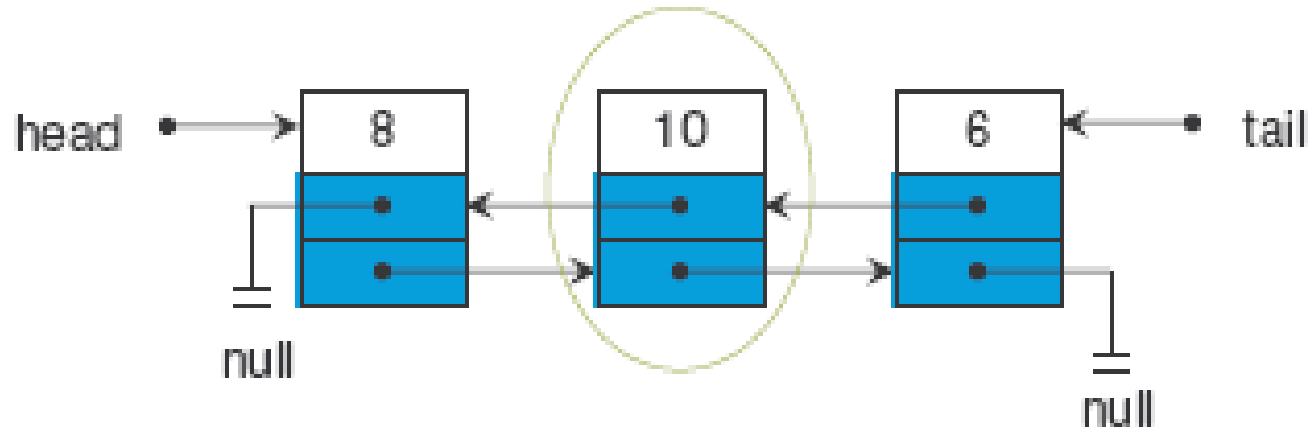
Add Last

3. tail= Baru

Add Last

```
void addLast(Node2P input){
    if (isEmpty()){
        head = input;
        tail = input;
    }
    else
    {
    input.previous = tail;
    tail.next = input;
    tail = input;
    } size++;
}
```
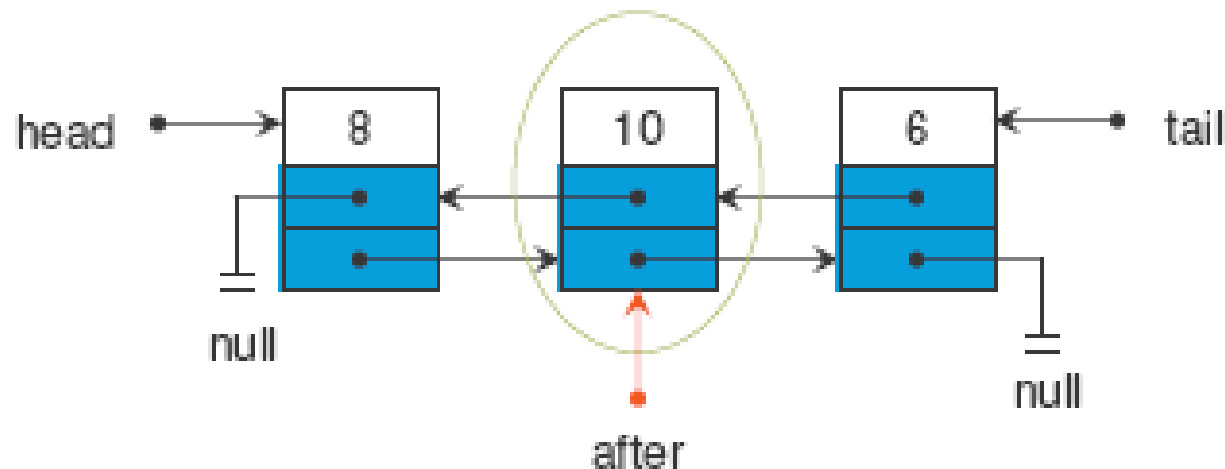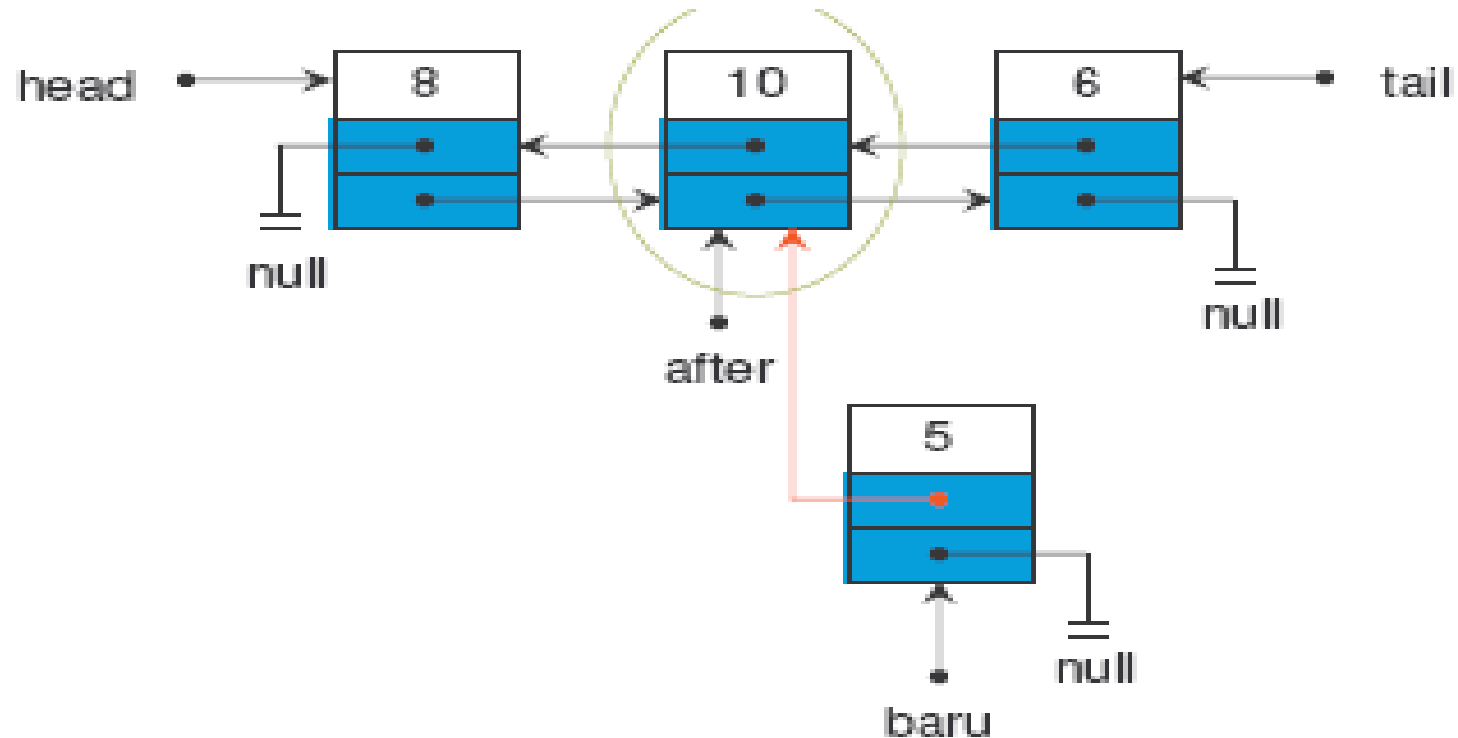
ALGORITMA DAN STRUKTUR DATA

Add After Node x

Add After Node x

We need a new reference **after.** Initialize **after** to **head** and shift it until the node x.
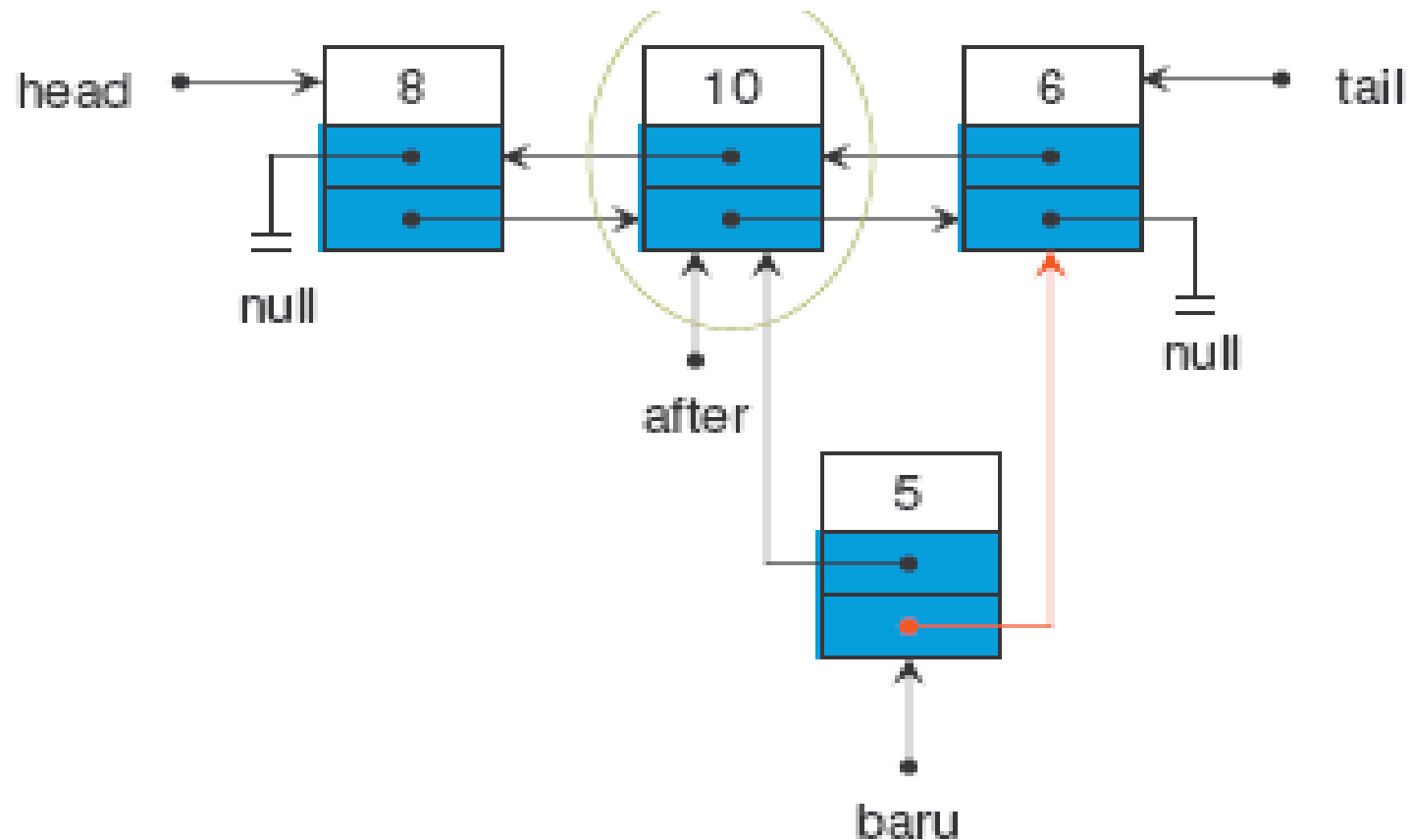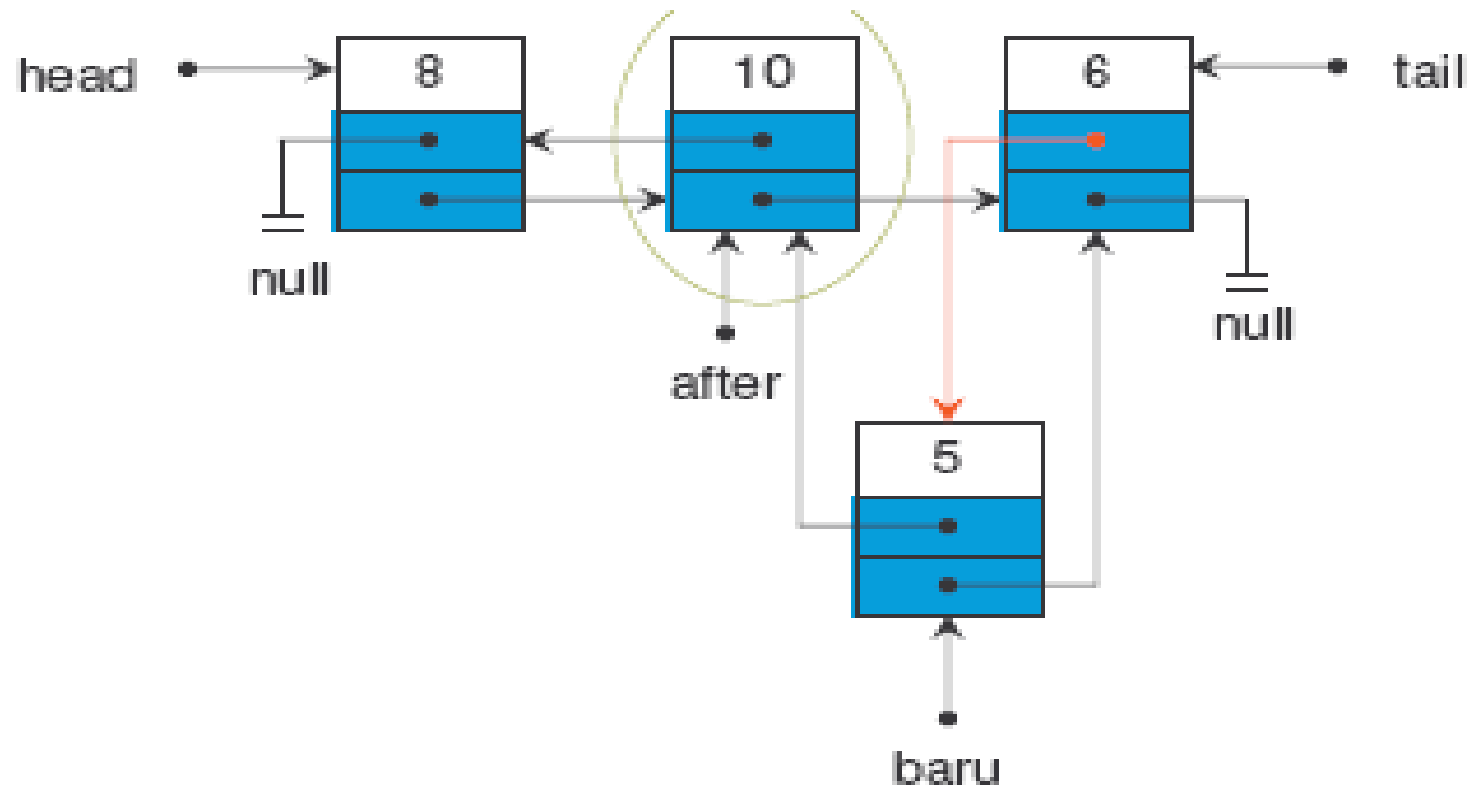
1. Node after = head;

ALGORITMA DAN STRUKTUR DATA

Add After Node x

2. Baru.prev = after

ALGORITMA DAN STRUKTUR DATA

Add After Node x

3. Baru.next = after.next

Add After Node x

4. after.next.prev = Baru

ALGORITMA DAN STRUKTUR DATA
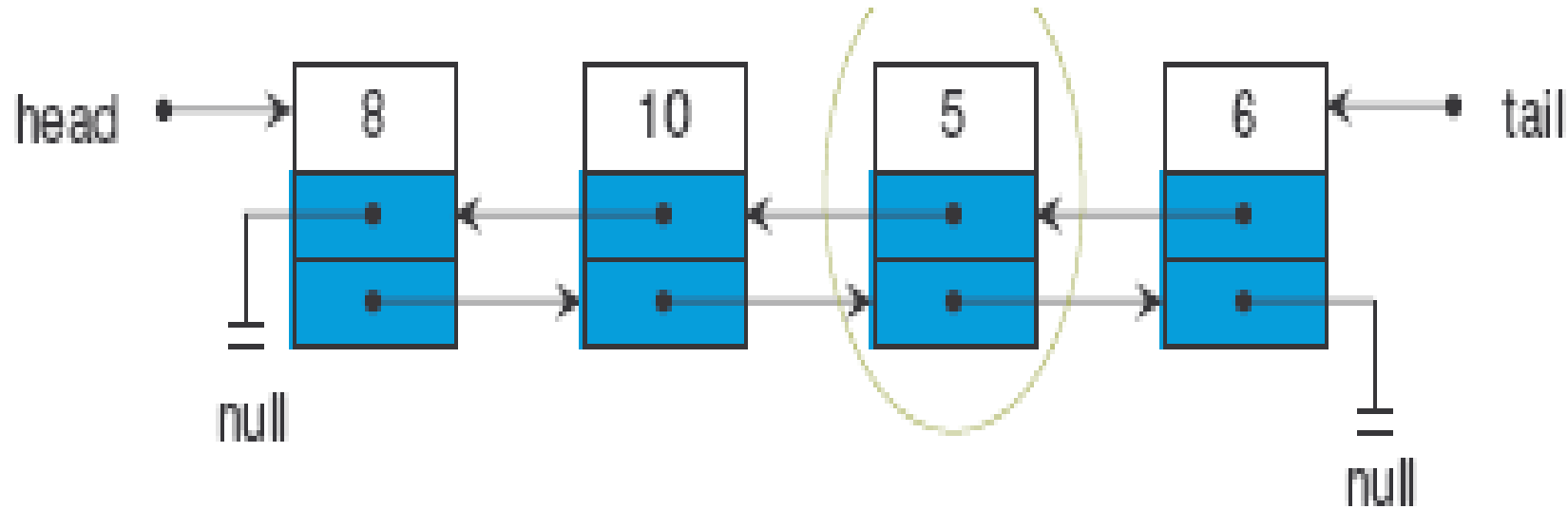
Add After Node x

5. after.next = Baru

ALGORITMA DAN STRUKTUR DATA
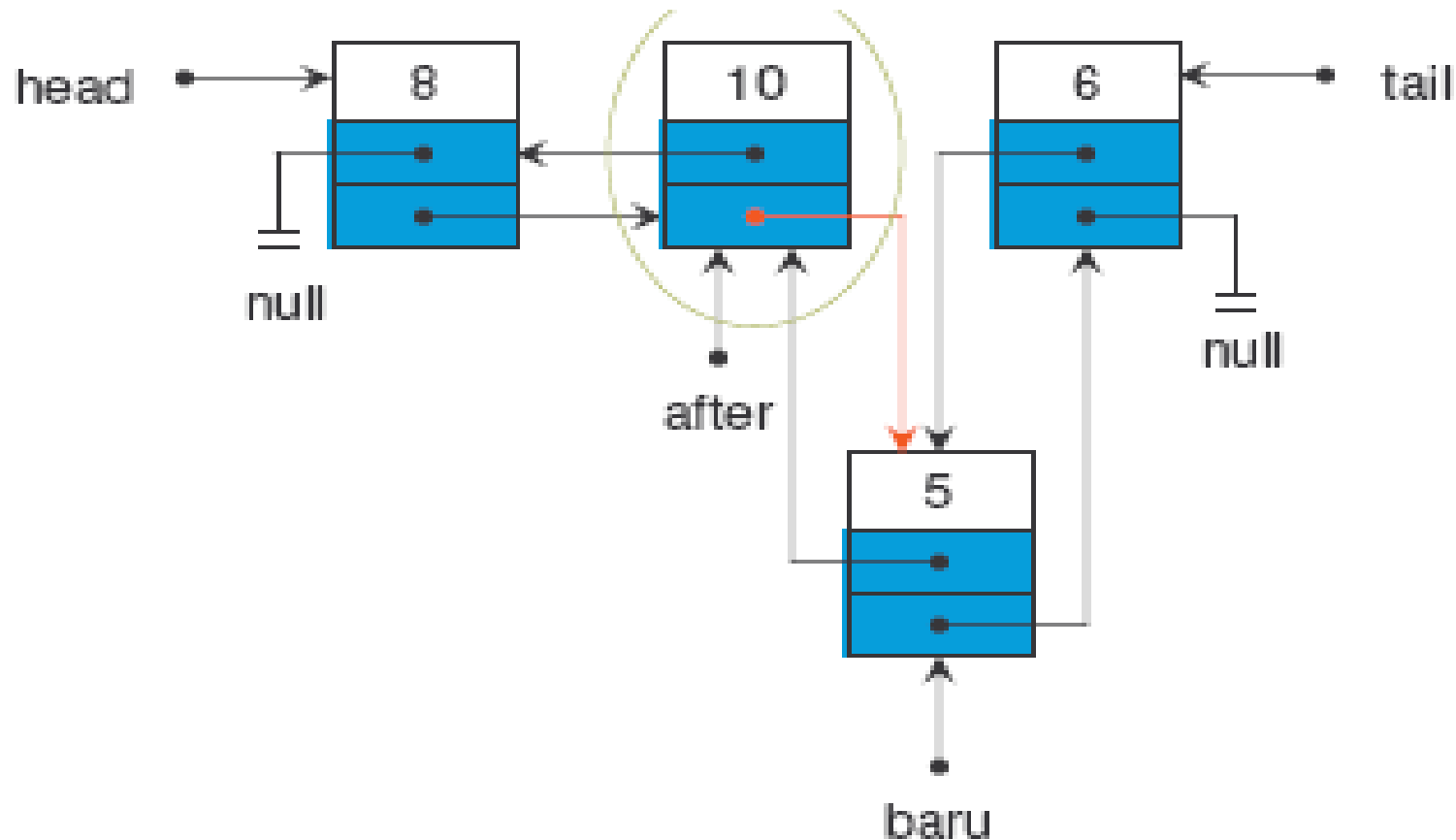
Add After Node x
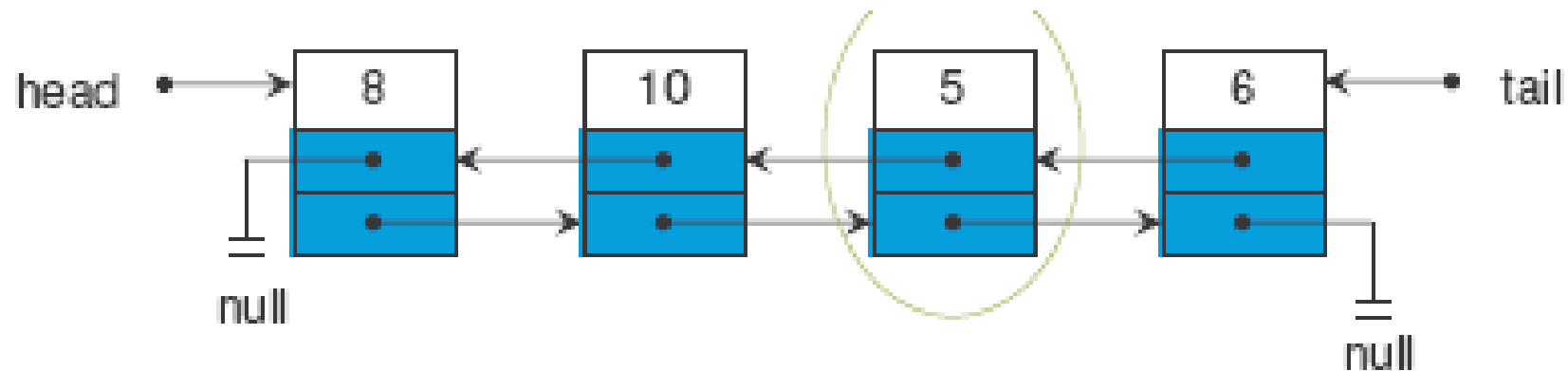
Final Result

Add After Node x

```java
void insertAfter(Object key,Node2P input){
    Node2P temp = head;
    do{
        if(temp.data==key){
            input.previous = temp;
            input.next = temp.next;
            temp.next.previous = input;
            temp.next = input;
            size++;
            System.out.println("Insert data is succeed.");
            break;
        }
        temp = temp.next;
    }while (temp!=null);
}
```
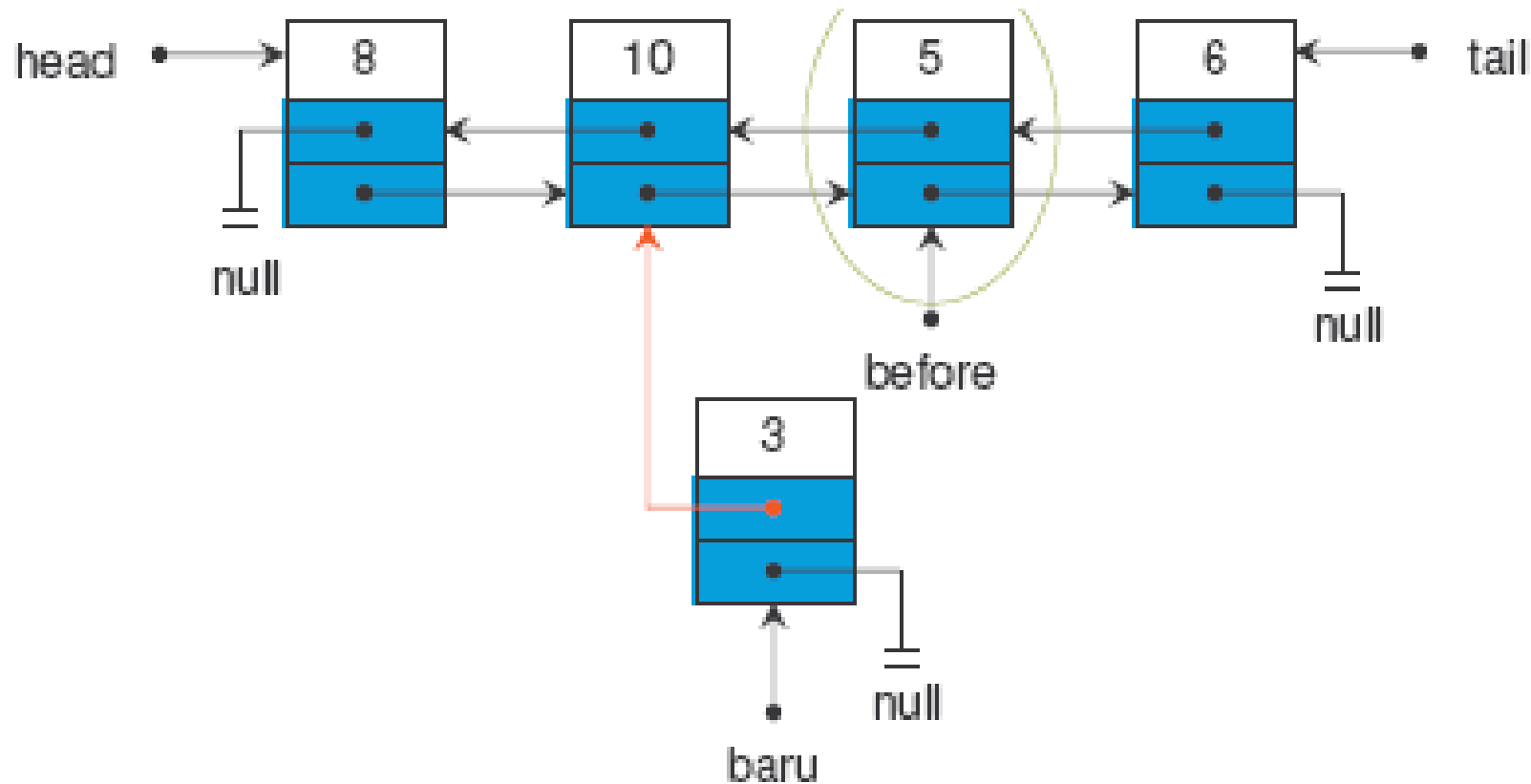
ALGORITMA DAN STRUKTUR DATA

Add Before Node x

Add Before Node x

1. Node **before** = head;

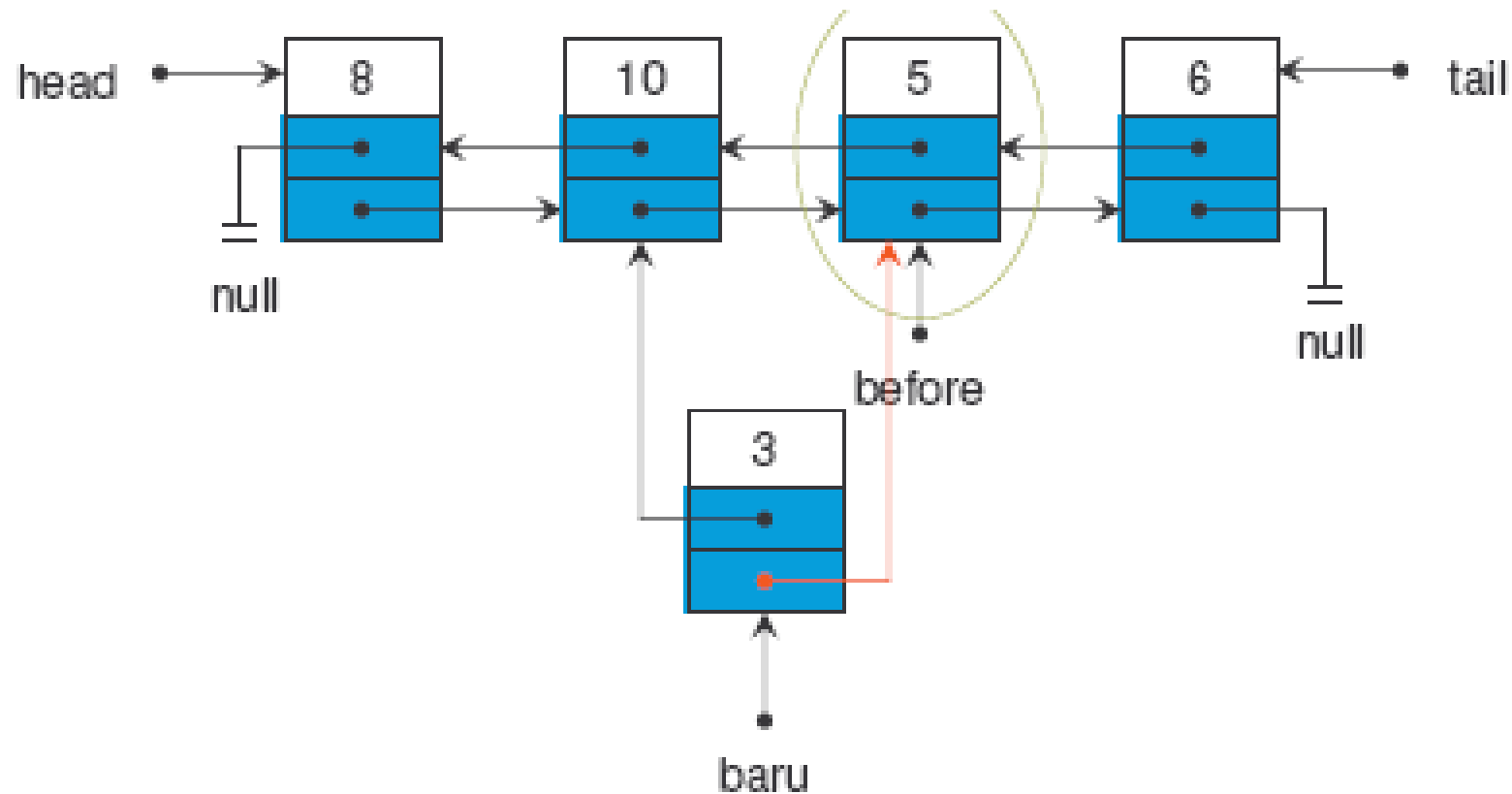   pointer **before** is firstly initialized to **head**, and it is then shifted to the correct location.
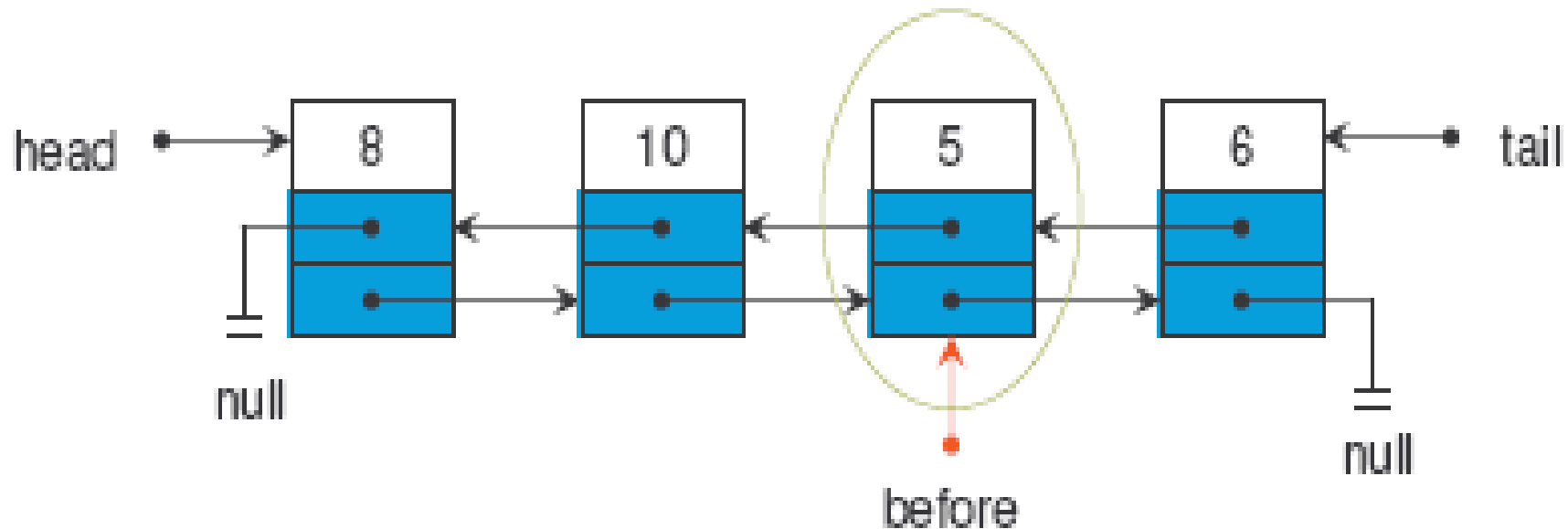
Add Before Node x
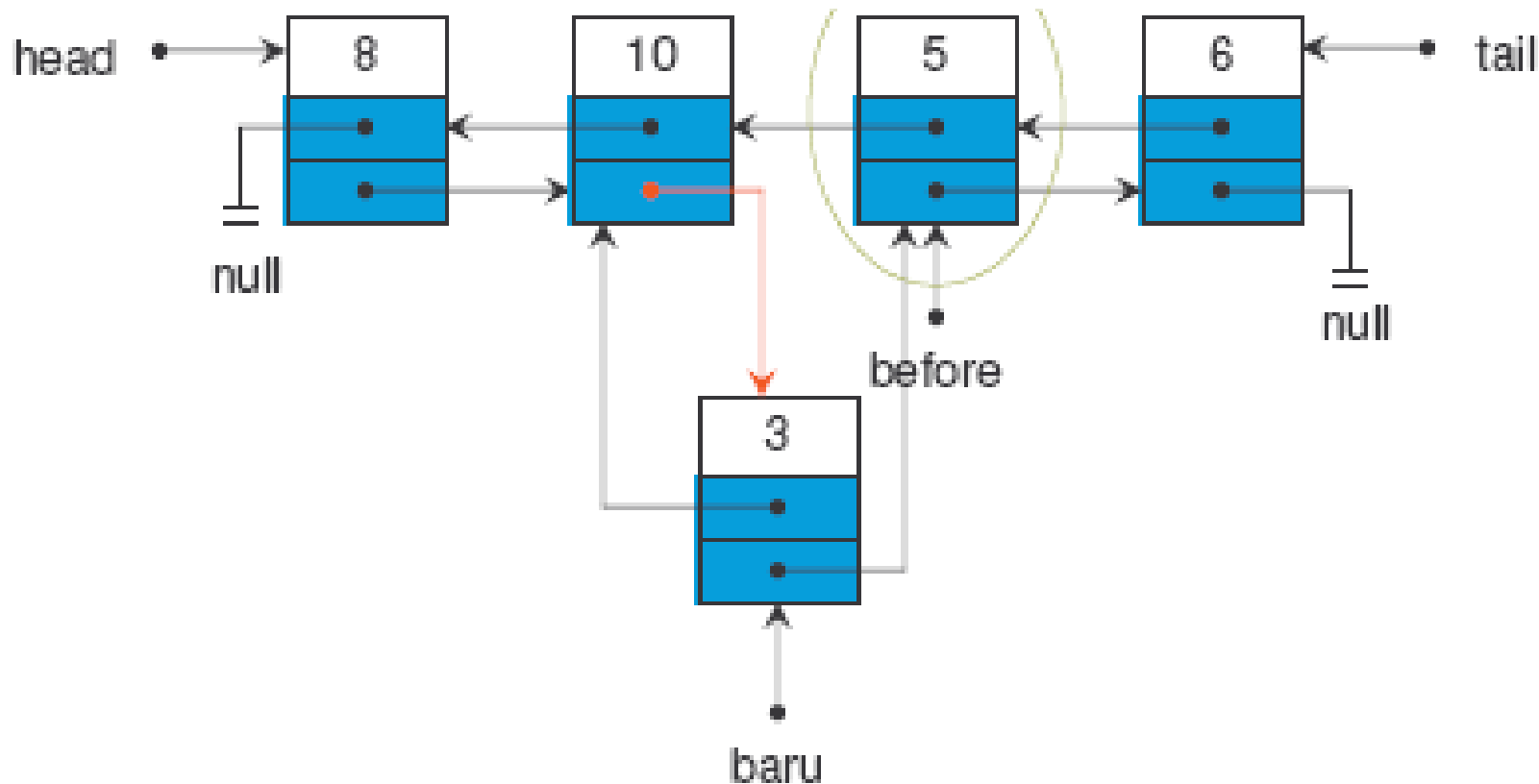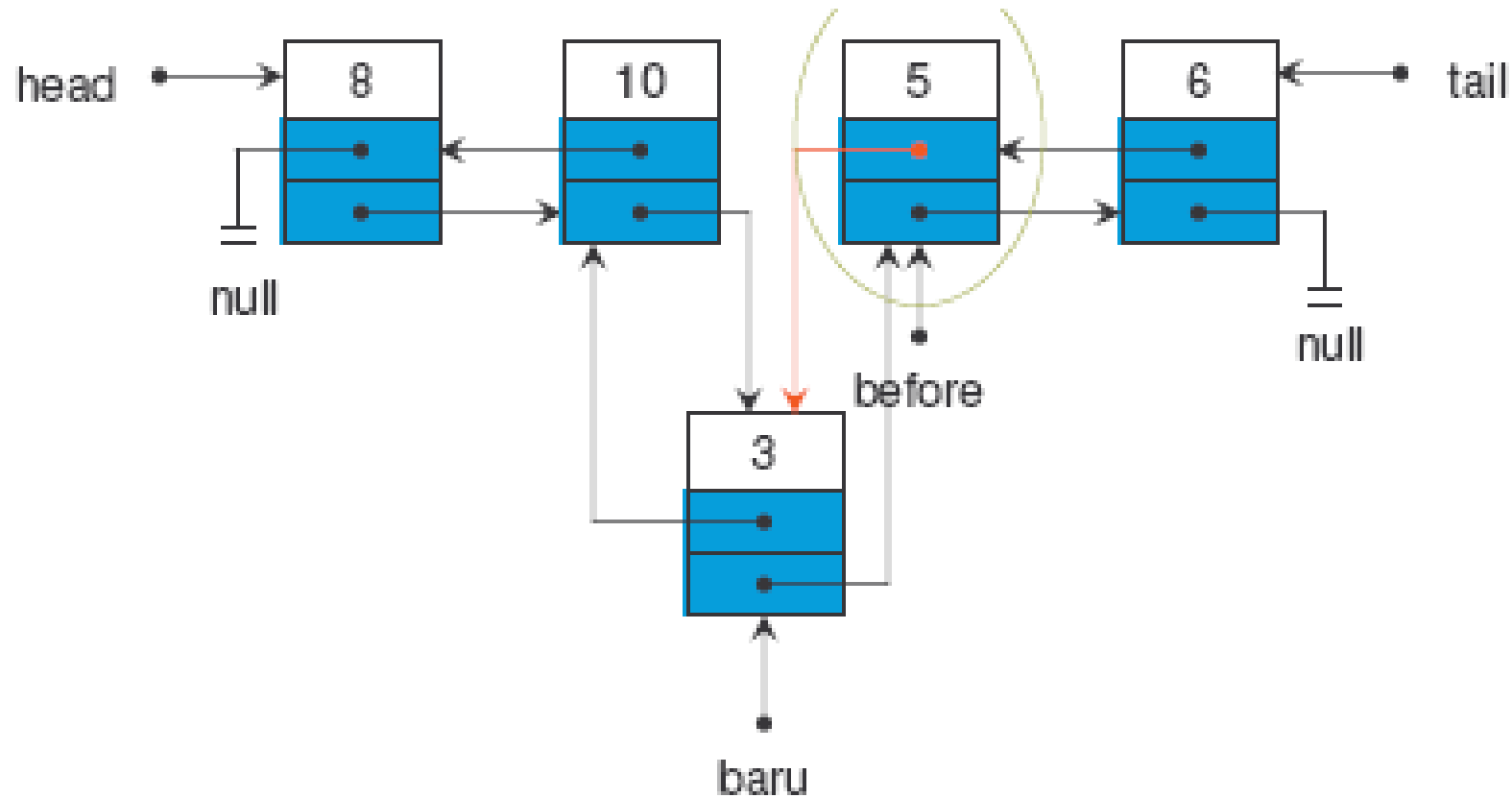
2. Baru.prev = before.prev

ALGORITMA DAN STRUKTUR DATA

Add Before Node x

3. Baru.next = before

ALGORITMA DAN STRUKTUR DATA

Add Before Node x

4. before.prev.next = Baru

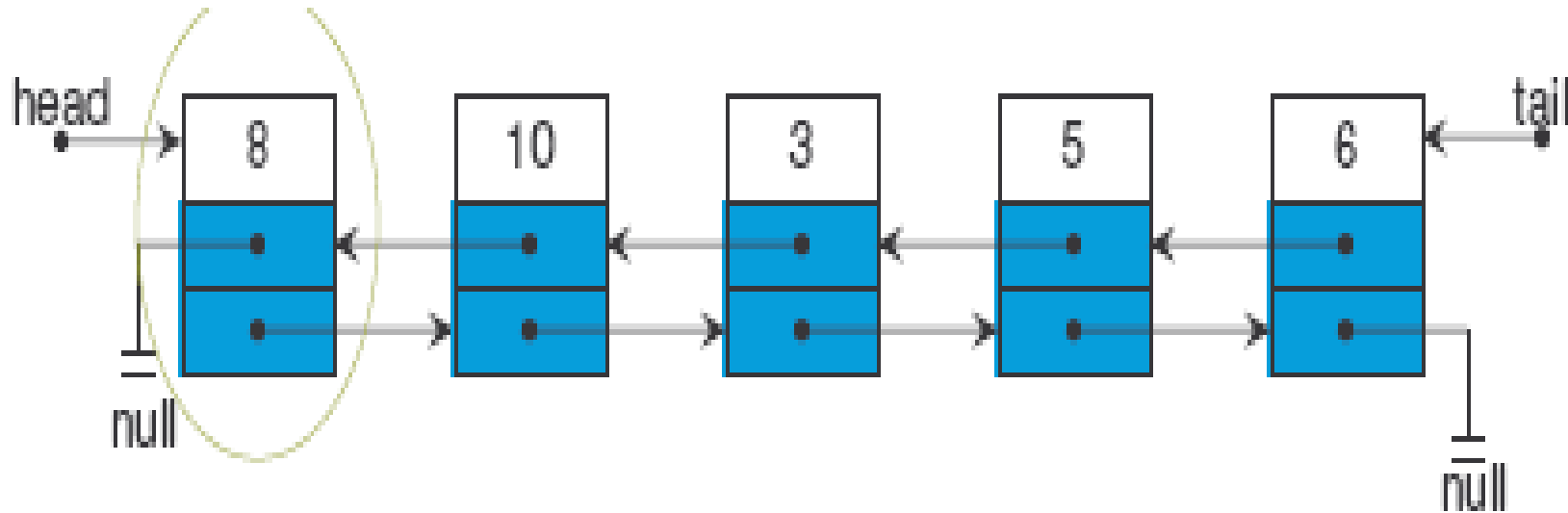ALGORITMA DAN STRUKTUR DATA

Add Before Node x

5. before.prev = Baru

Add Before Node x

Final Result:

# Add Before Node x

```java
void insertBefore(Object key,Node2P input){
    Node2P temp = head;
    while (temp != null){
        if (temp.data == key)
        {
            if(temp == head)
            {
                this.addFirst(input);
                System.out.println("Insert data is succeed.");
                size++;
                break;
            }
            else
            {
                input.previous = temp.previous;
                input.next = temp;
                temp.previous.next = input;
                temp.previous = input;
                System.out.println("Insert data is succeed.");
                size++;
                break;
            }
        }
        temp = temp.next;
    }
```
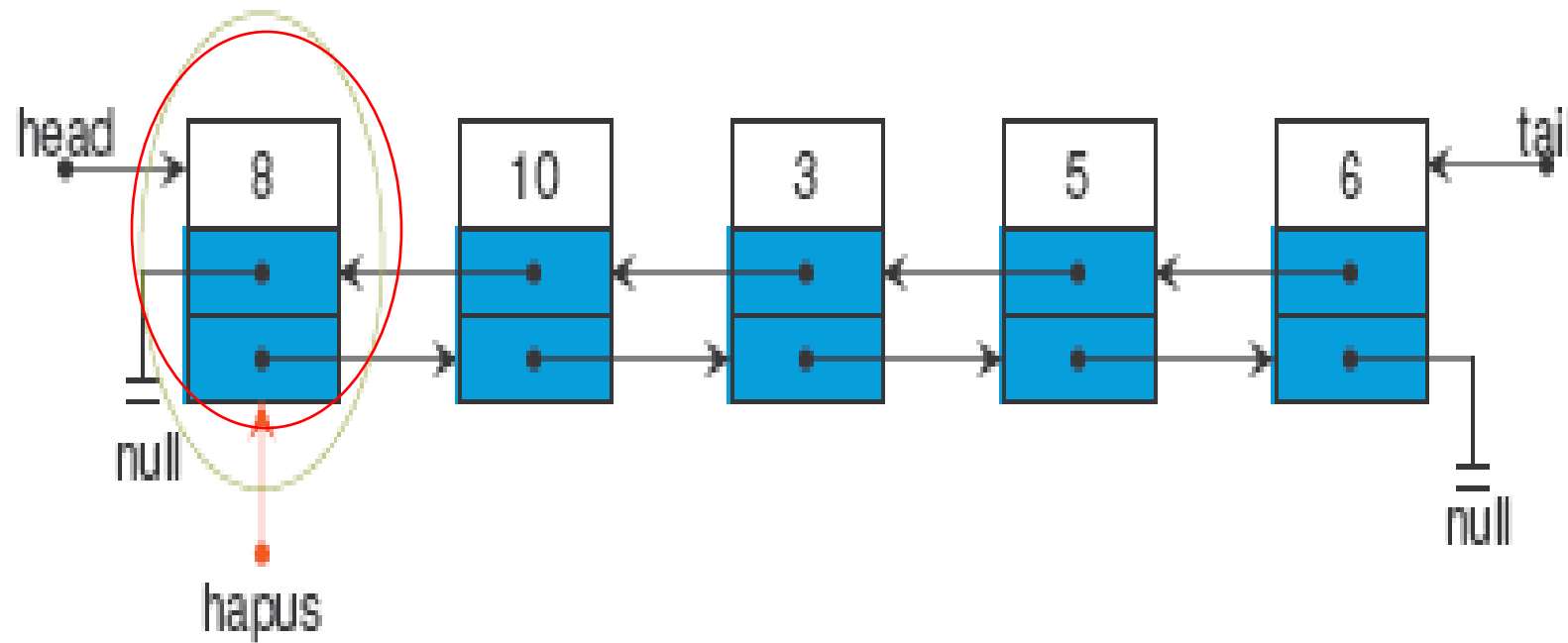
(5) Deletion

- Dibedakan menjadi :
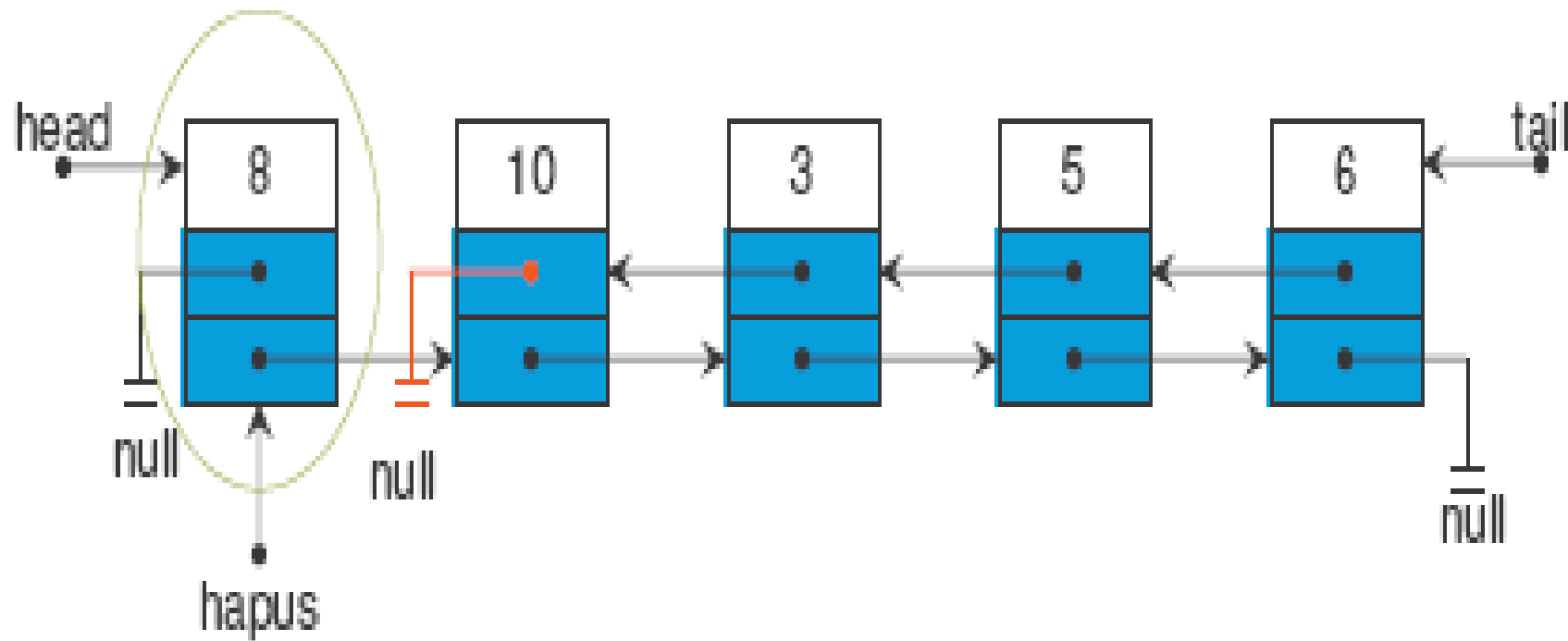1. Delete First
2. Delete Last
3. Delete

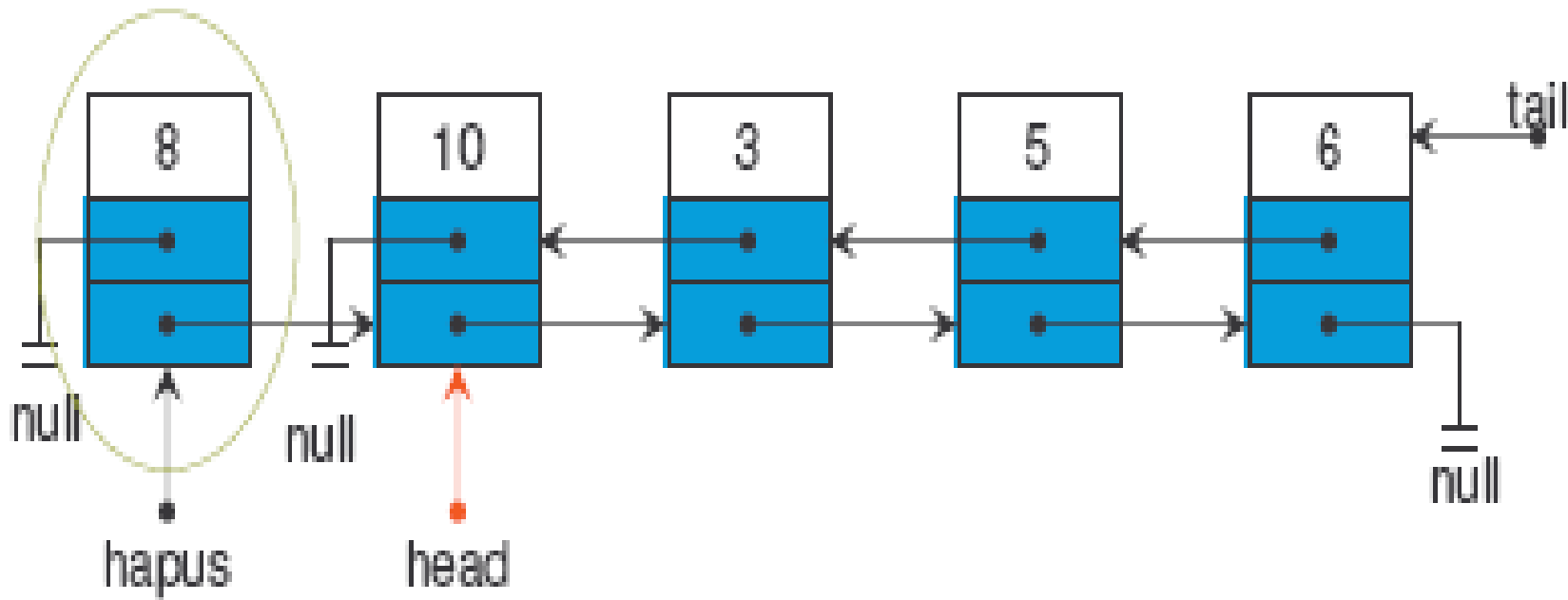# Delete First
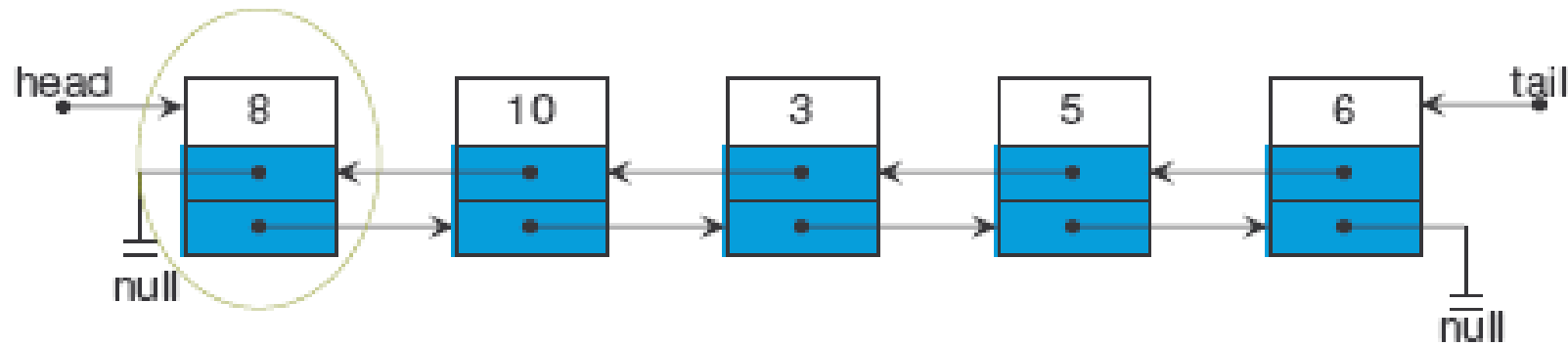
Delete First

1. Node hapus = head;
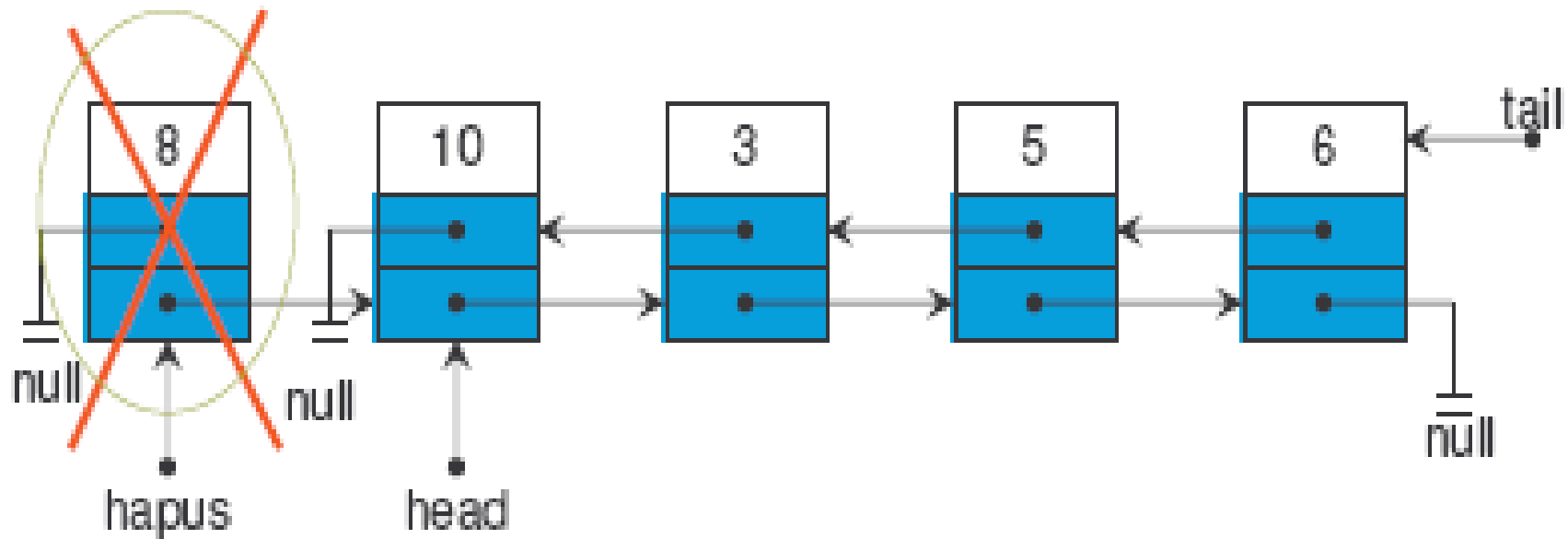
Delete First

2. head.next.prev = null

ALGORITMA DAN STRUKTUR DATA

Delete First

3. head = hapus.next
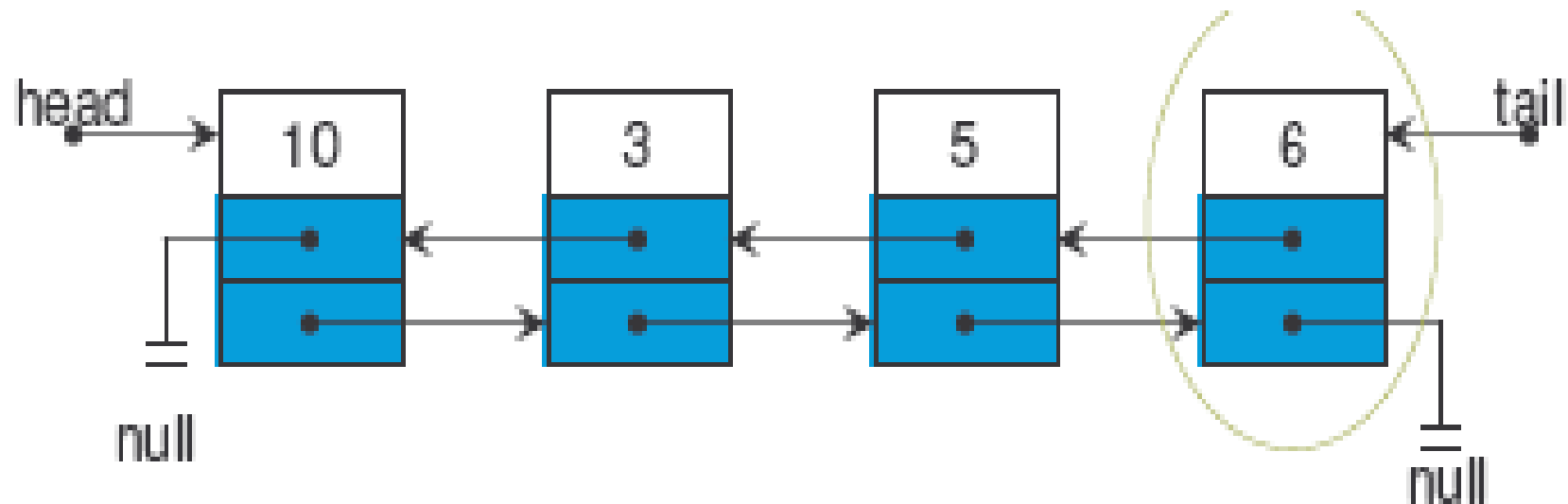
Delete First

Node 8 sudah terhapus

Delete First
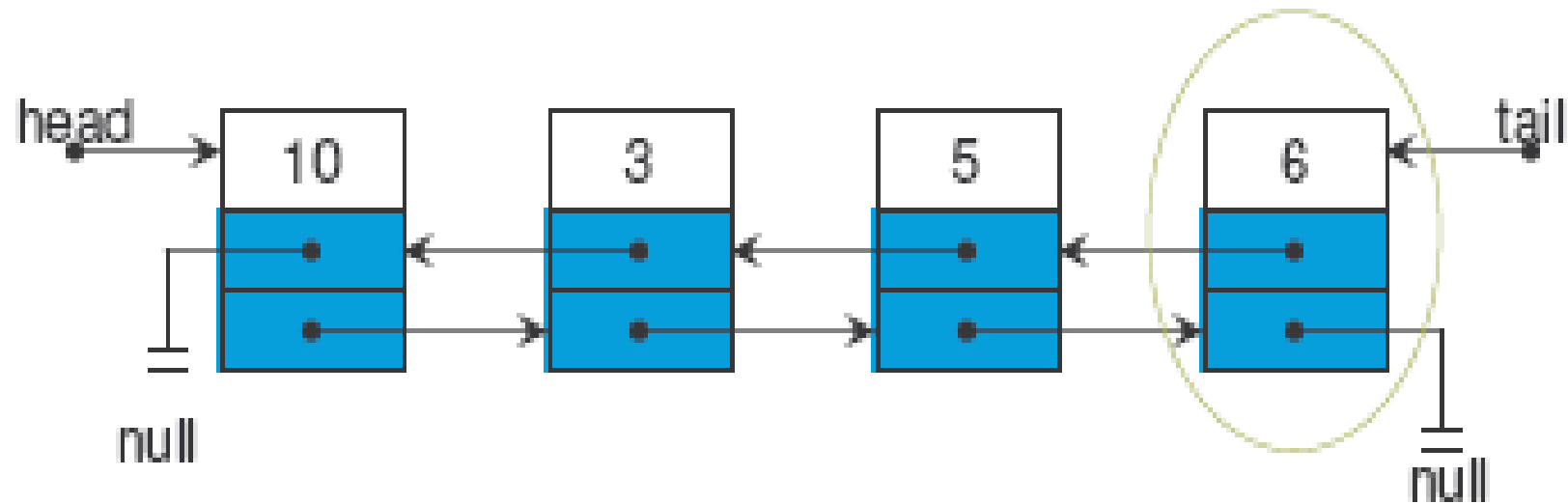
Hasil akhir :

Delete First

```
void removeFirst(){
    Node2P temp = head;
    if (!isEmpty()){
        if (head == tail)
            head = tail = null;
        else
        {
            head.next.previous = null;
            head = temp.next;
        } size--;
    }
    else
        System.out.println("Data is empty!");

}
```
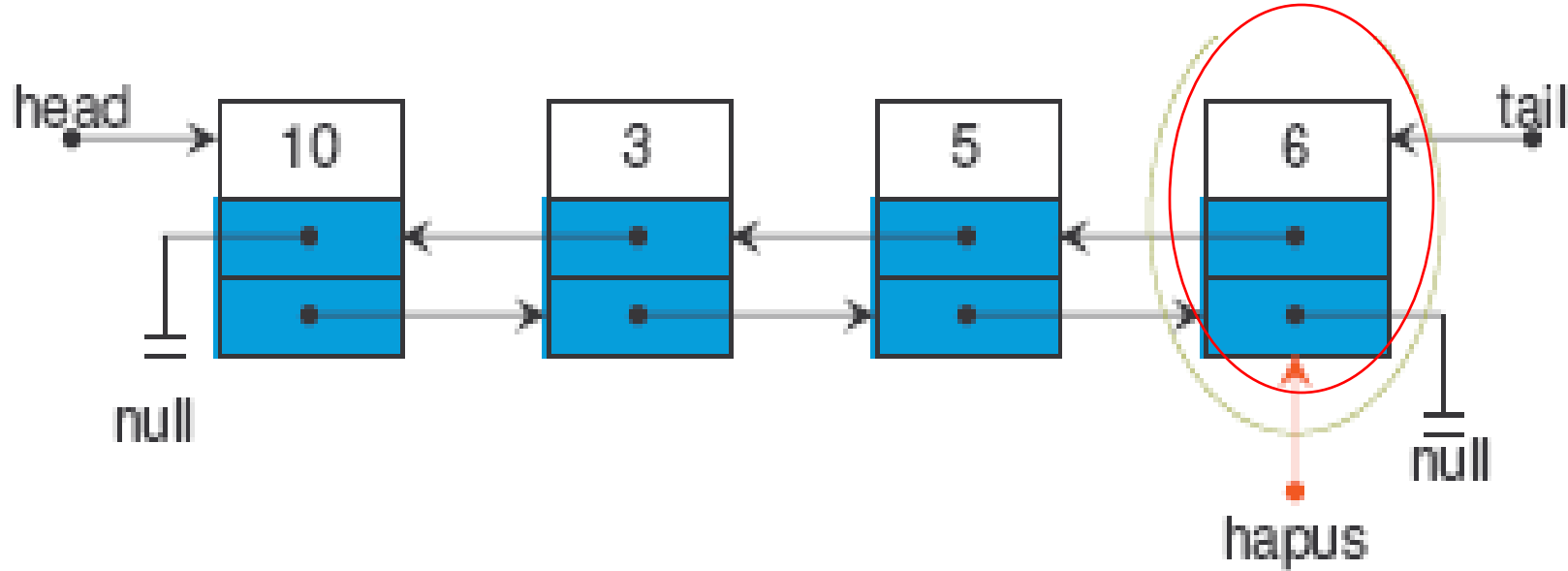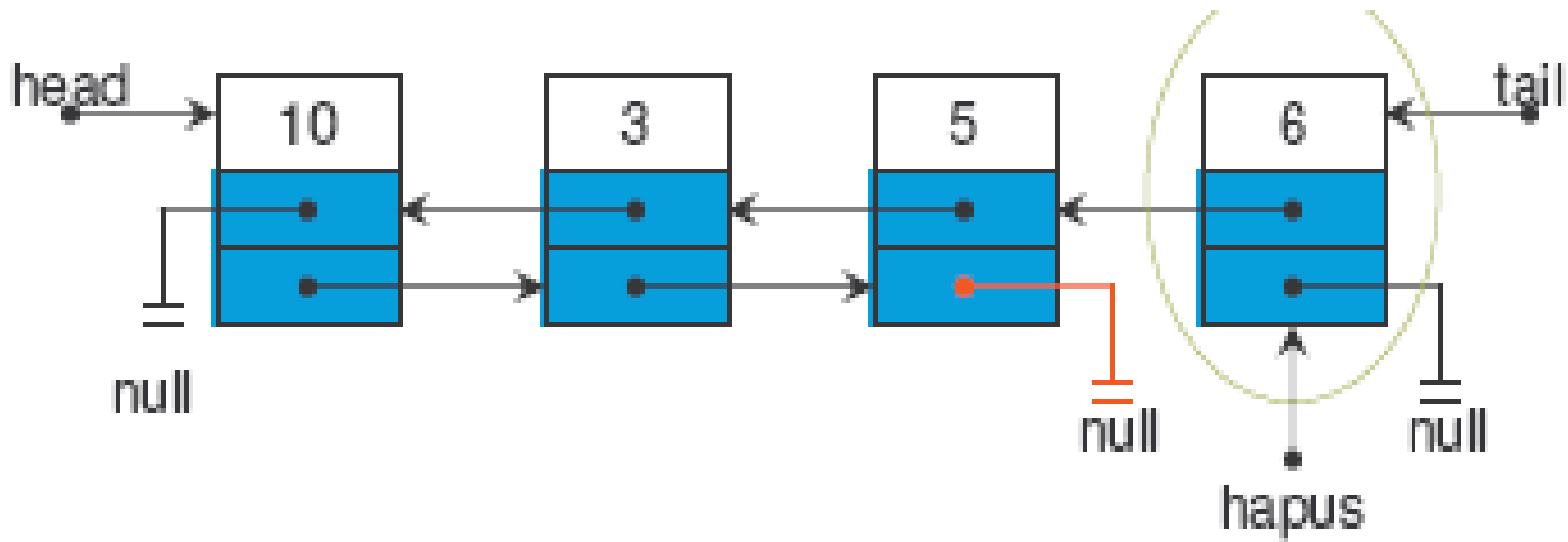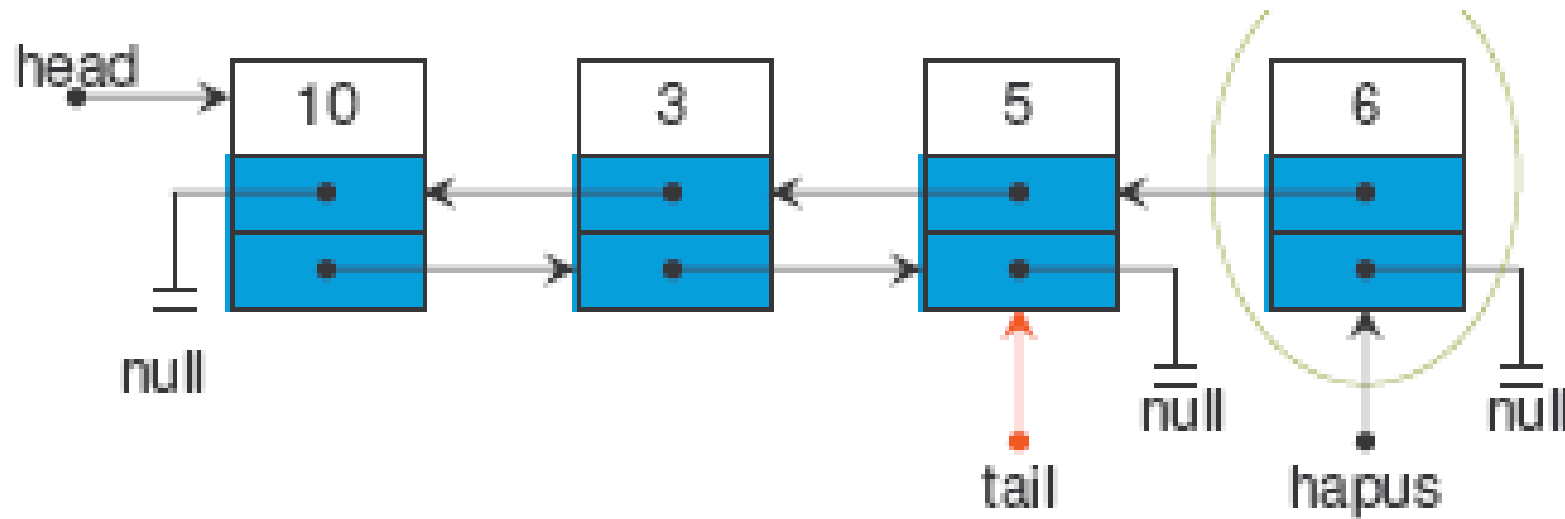
Delete Last

ALGORITMA DAN STRUKTUR DATA

Delete Last

1. Node hapus=tail;

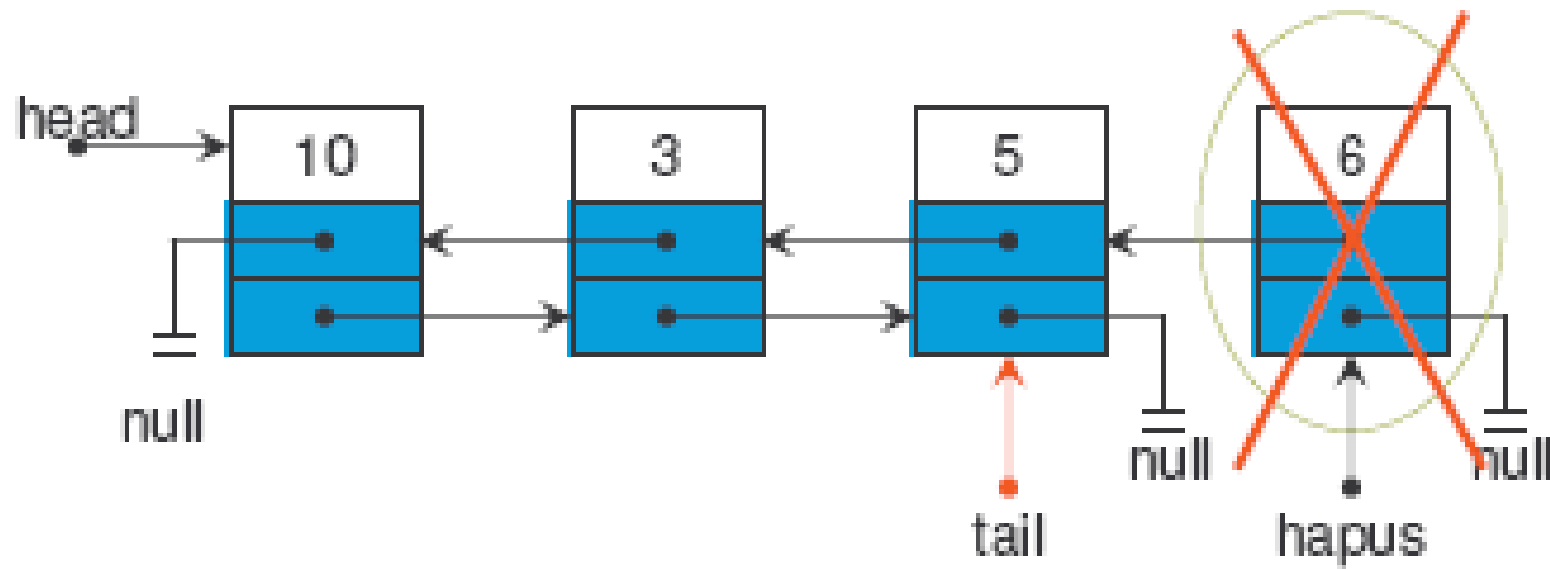ALGORITMA DAN STRUKTUR DATA

Delete Last

2. tail.prev.next = null

Delete Last

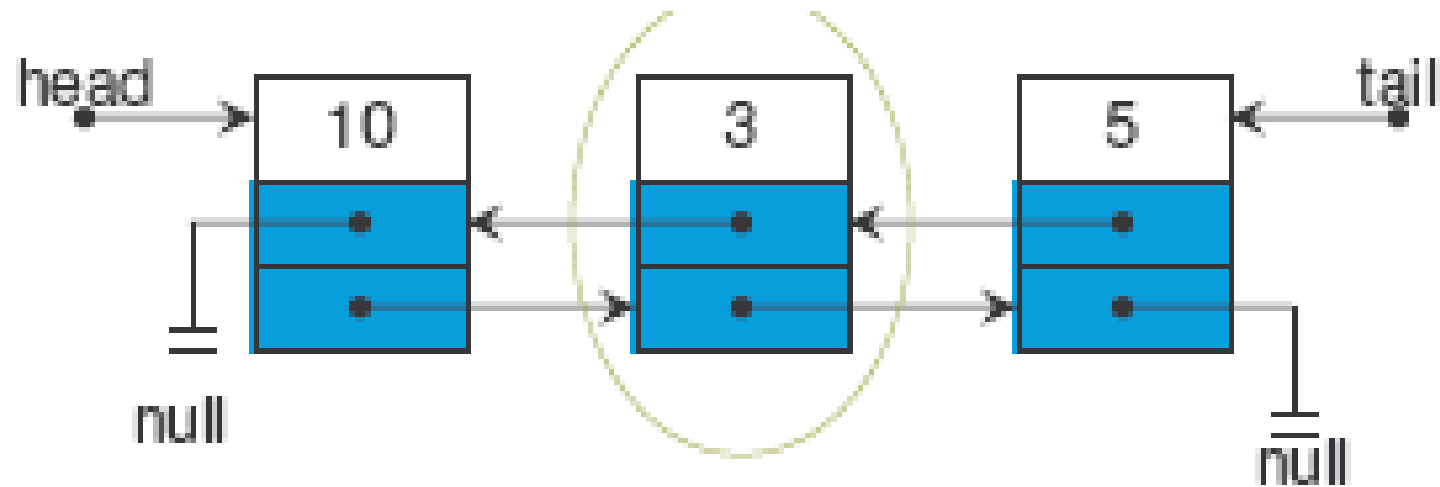3. tail= hapus.prev

Delete Last

Node 6 sudah terhapus

Delete Last

Delete Last

```
void removeLast(){
    Node2P temp = tail;
    if (!isEmpty()){
        if (tail == head){
            head = tail = null;
        }
        else {
            tail.previous.next = null;
            tail=temp.previous;
        } size--;
    }
    else System.out.println("Data is empty!");
}
```

Delete Node x

```java
void remove(Object key){
    Node2P temp = head;
    if (!isEmpty()){
        while (temp != null){
            if (temp.data == key){
                if (temp == head){
                    this.removeFirst();
                    size--;
                    break;
                }
                else
                {
                    temp.previous.next = temp.next;
                    temp.next.previous = temp.previous;
                    if(temp.next == null)
                        tail=temp;
                    size--;
                    break;
                }
            }
            temp = temp.next;
        }
    }
    else
        System.out.println("Data is empty!");
    size--;
}
```
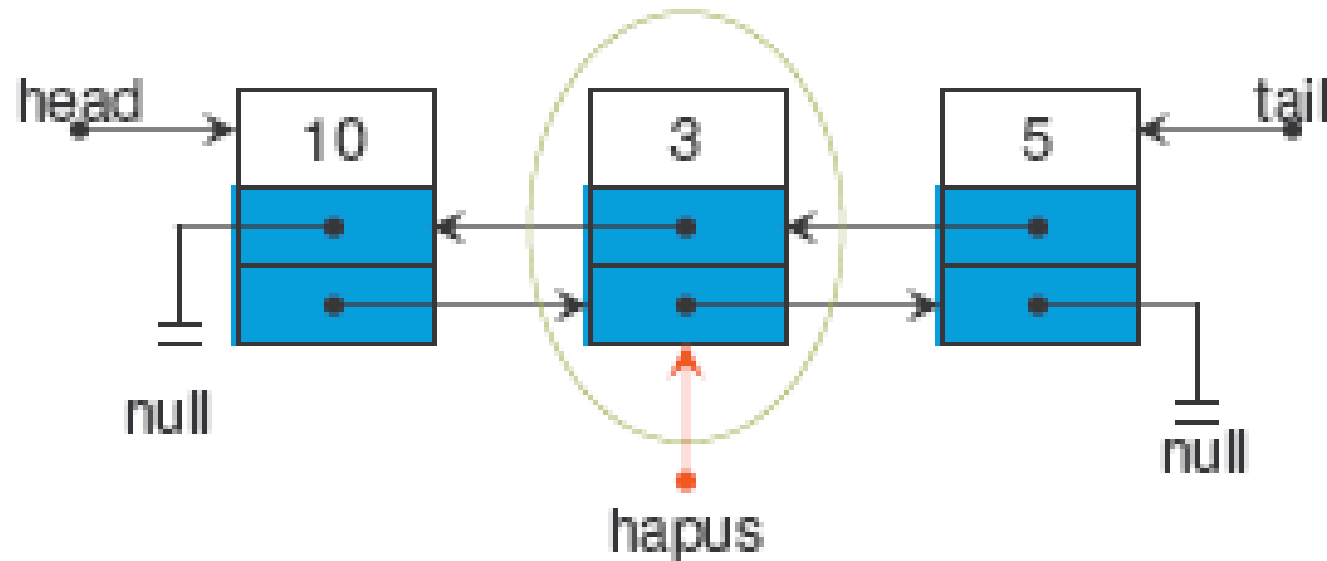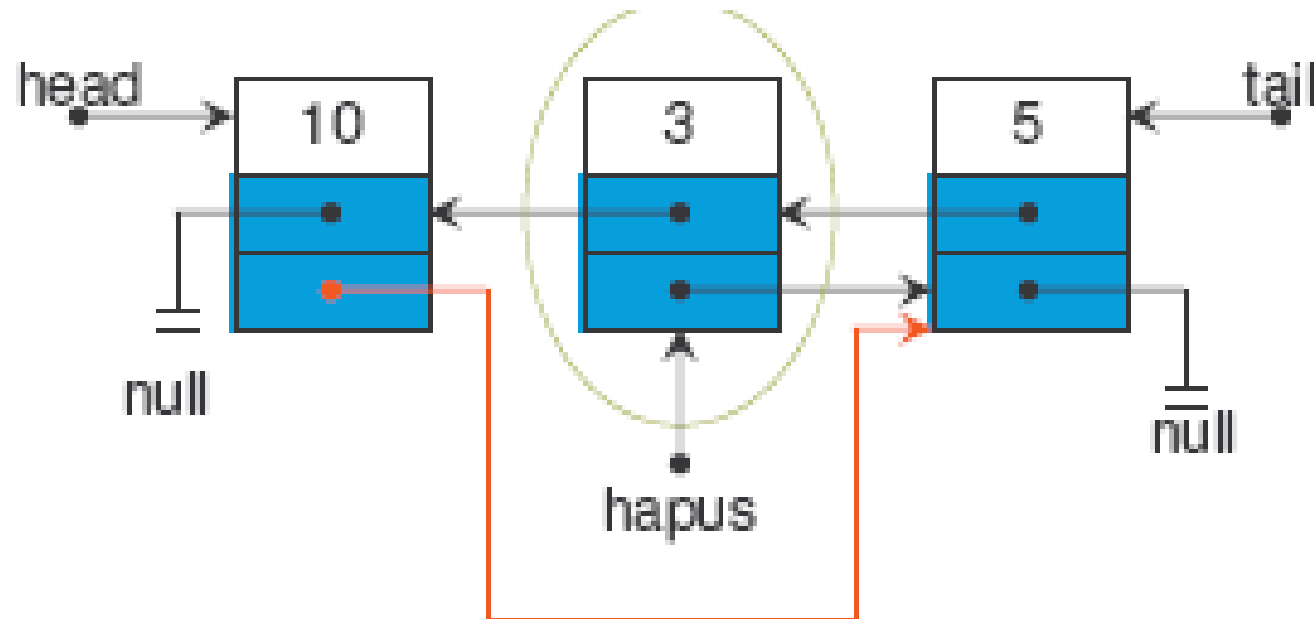
Delete Node x
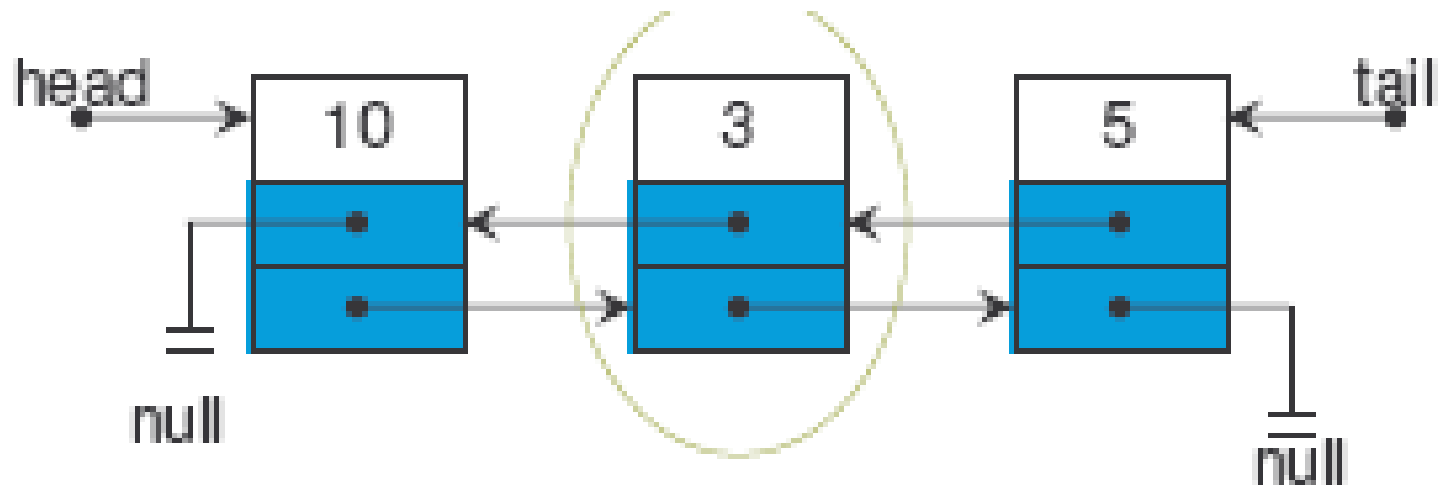
Delete Node x

1. Node hapus=head;

   Initialize **hapus** as **head**. Through traversal process, move **hapus** to node x.
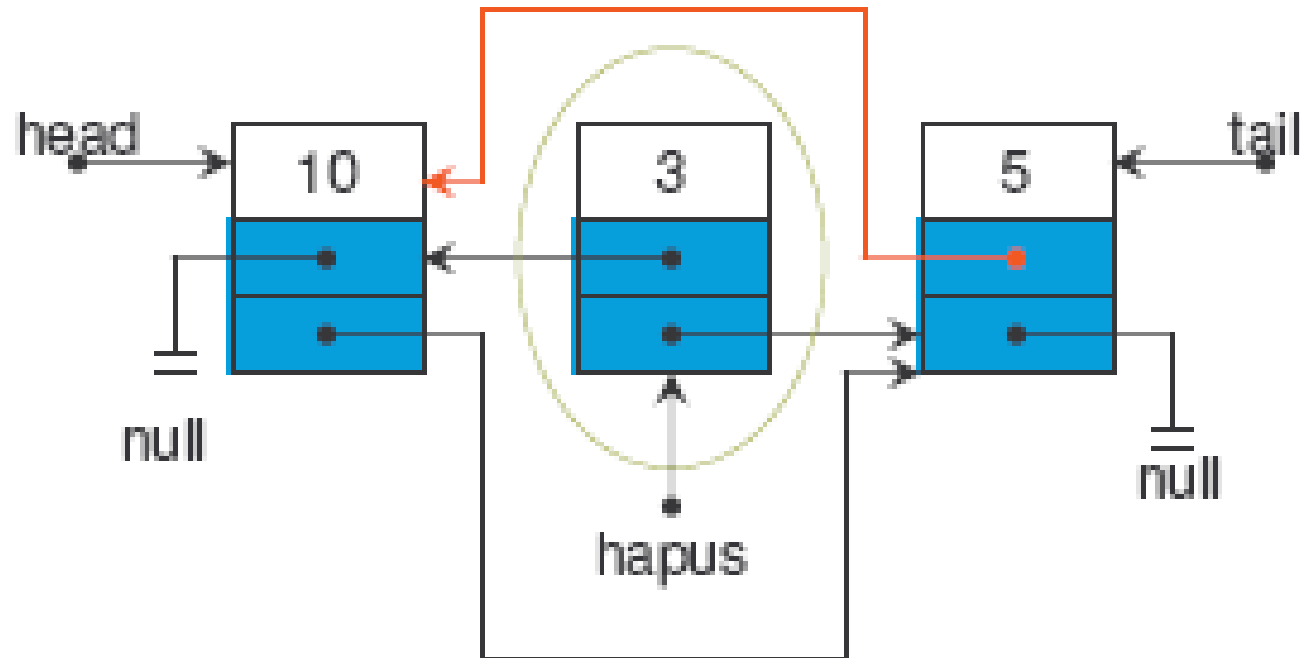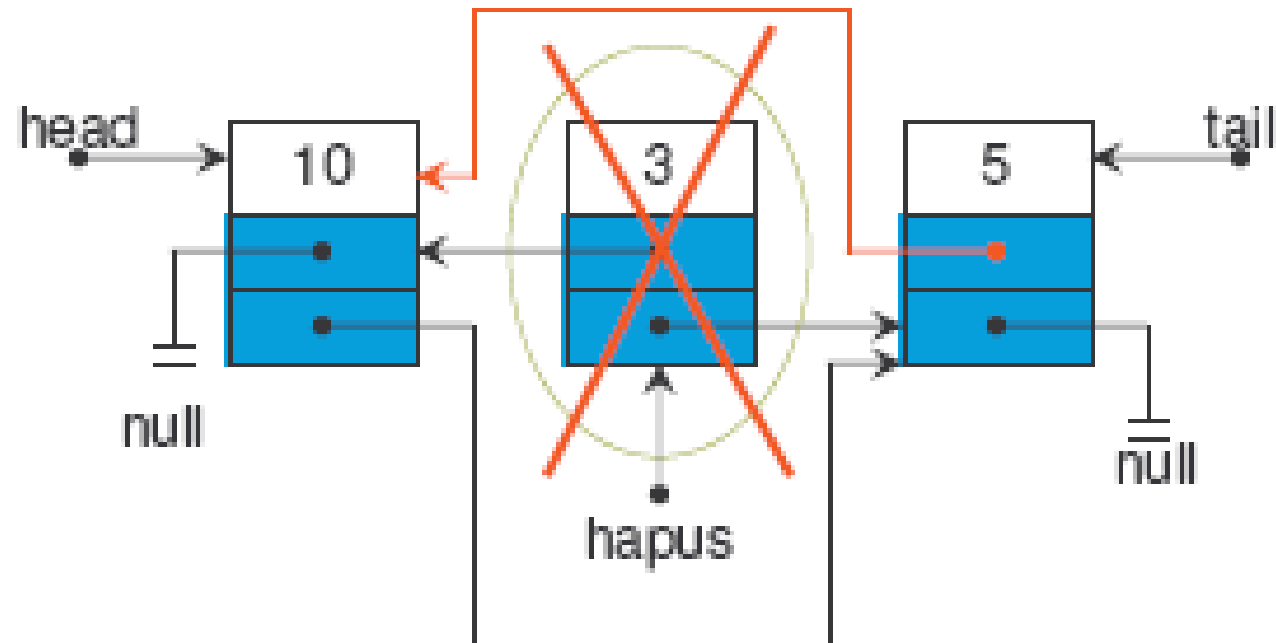
Delete Node x

2. hapus.prev.next = hapus.next;
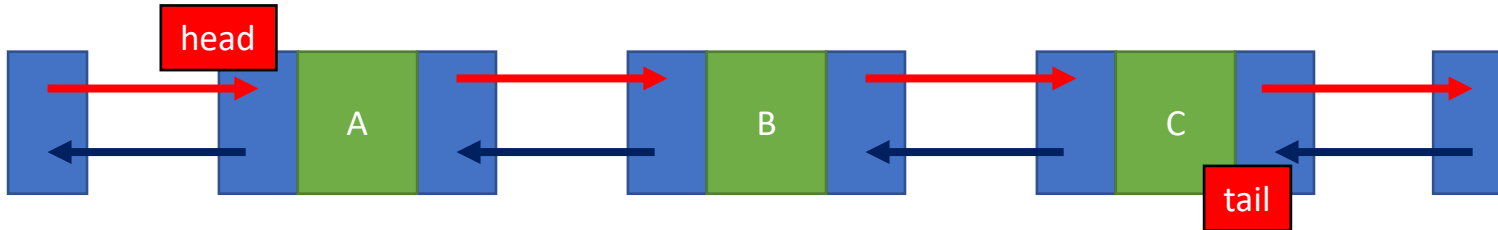
Delete Node x

3. hapus.next.prev = hapus.prev;

Delete Node x

Assignments

Explain the steps of the following cases, implemented to the given linked list!

1. Add new node D before B.
2. Add new node E after C.
3. Add new node F after D.
4. Add new node G at the 3rd position.
5. Add new node H at the first position.
6. Add new node I at the second position.
7. Delete the first node
8. Delete the last node
9. Delete A node
10. Delete node at 5th position
11. Display all nodes

ALGORITMA DAN STRUKTUR DATA

Thank you ☺