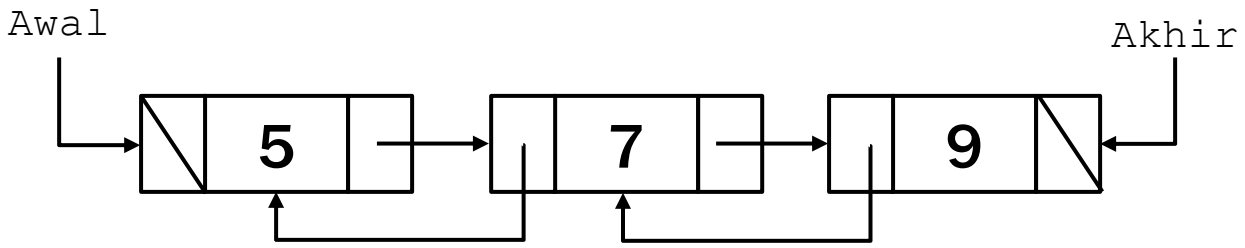
DOUBLE LINKED LIST

OLEH : ANDRI HERYANDI, M.T.

DEFINISI DOUBLE LINKED LIST

- Double linked list adalah linked list yang setiap nodenya memiliki 2 field link. 1 field menunjuk ke node berikutnya dan 1 field lain menunjuk ke field sebelumnya.
- Double linked list digunakan ketika anda membutuhkan linked list yang mempunyai kemampuan pindah node secara maju (menggunakan field next) atau pun mundur (menggunakan field previous/prev).
- Secara ukuran di memori, double linked list membutuhkan memori lebih banyak karena ada tambahan field untuk setiap nodenya.



STRUKTUR NODE DOUBLE LINKED LIST

- Struktur node pada double linked list adalah sebagai berikut

- Field **Prev** menyimpan link/pointer yang menunjuk ke node sebelumnya.
- Field **Informasi** menyimpan data yang dimiliki oleh node tersebut.
- Field **Next** menyimpan link/pointer yang menunjujk ke node selanjutnya.

STRUKTUR NODE DOUBLE LINKED LIST

- Pendeklarasian Class Node Bahasa Python adalah :

```
class Node:
    def __init__(self, info):
        self.info = info
        self.next = None
        self.prev = None
```

- Pendeklarasian Variable/Object Node :

```
node1 = Node(10)
```

Object dengan nama node1 berupa node yang mempunyai nilai/info berupa bilangan bulat dengan nilai 10

STRUKTUR NODE DOUBLE LINKED LIST

- Pendeklarasian Class Node Bahasa Python adalah :

```
class Node:
    def __init__(self, info):
        self.info = info
        self.next = None
        self.prev = None
```

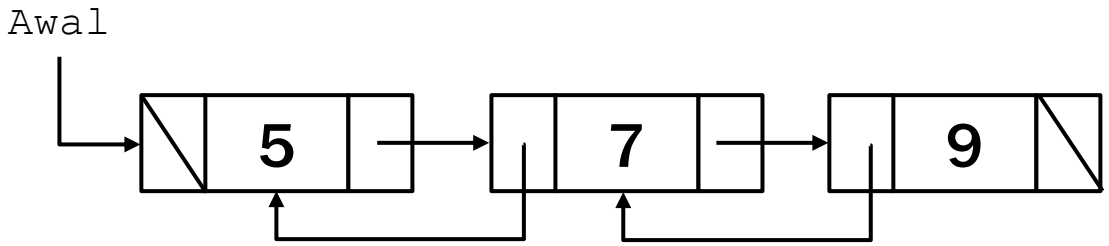
- Pendeklarasian Variable/Object Node :

```
node1 = Node(10)
```

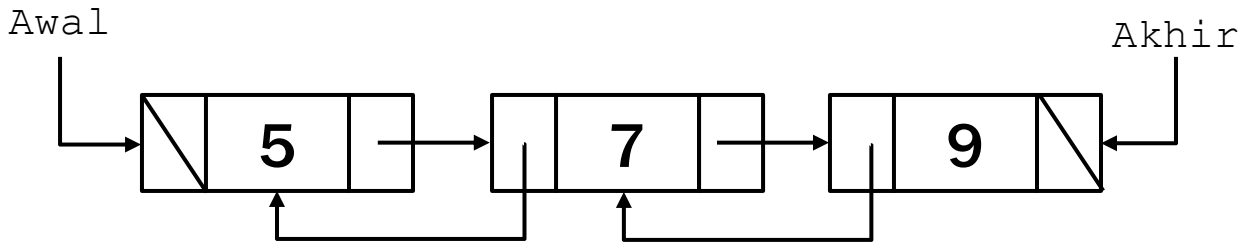
Object dengan nama node1 berupa node yang mempunyai nilai/info berupa bilangan bulat dengan nilai 10

STRUKTUR DOUBLE LINKED LIST

- Sebuah double linked list dapat dibuat dengan sebuah pointer yang menunjuk ke node awal dari linked list saja.



- Tetapi boleh juga anda membuat sebuah double linked list yang mempunyai 2 penunjuk yaitu ke node awal dan node akhir. Linked list ini mempunyai kelebihan yaitu mudahnya mengakses node akhir. Tetapi linked list ini juga mempunyai kerumitan lain karena kita harus melibatkan lebih banyak proses dalam operasinya.



STRUKTUR NODE DOUBLE LINKED LIST

- Dalam perkuliahan ini, double linked list yang akan digunakan adalah double linked list yang menggunakan 2 penunjuk (awal dan akhir). Pendeklarasian Class Linked List dalan Bahasa Python adalah :

```
class DoubleLinkedList:
    def __init__(self):
        self.awal = None
        self.akhir  = None
```

- Pendeklarasian sebuah Variable/Object Double Linked List adalah :

```
list1 = DoubleLinkedList()
```

OPERASI-OPERASI DALAM DOUBLE LINKED LIST (DIBANDINGKAN DENGAN SINGLE LINKED LIST)

- Inisialisasi linked list [harus ada modifikasi]
- Pembuatan Node Baru [harus ada modifikasi]
- Pemeriksaan kondisi kosong linked list. [sama saja dengan single linked list]
- Traversal Linked List
  - Traversal untuk menampilkan linked list dari node awal menuju node akhir [sama saja dengan single linked list]
  - Traversal untuk menampilkan linked list dari node akhir menuju node awal. [dibuat baru]
  - Traversal untuk menghitung banyaknya elemen linked list [sama saja dengan single linked list]
- Pencarian node berdasarkan posisi
  - Mencari node di posisi pertama [harus ada modifikasi]
  - Mencari node di posisi tertentu [harus ada modifikasi]
  - Mencari node di posisi akhir [harus ada modifikasi]

OPERASI-OPERASI DALAM DOUBLE LINKED LIST

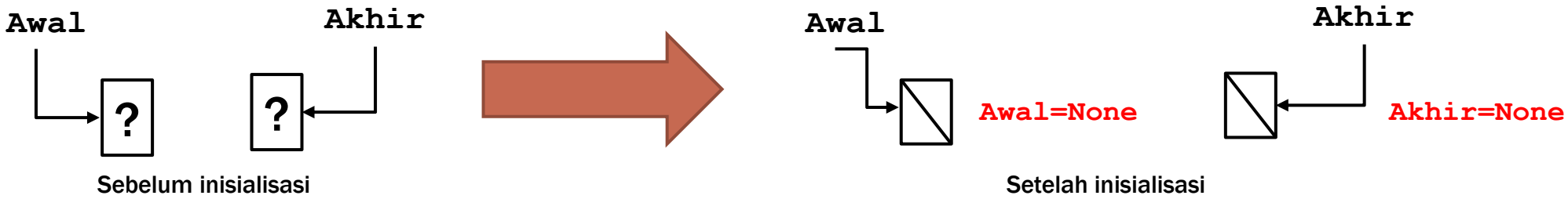
- Penambahan Data
  - Penambahan di awal linked list [harus ada modifikasi]
  - Penambahan di tengah (penyisipan) linked list [harus ada modifikasi]
  - Penambahan di akhir linked list [harus ada modifikasi]
- Update Data [sama saja dengan single linked list]
- Penghapusan Data
  - Penghapusan di awal linked list [harus ada modifikasi]
  - Penghapusan di tengah linked list [harus ada modifikasi]
  - Penghapusan di akhir linked list [harus ada modifikasi]
  - Penghapusan semua elemen [sama saja dengan single linked list]

INISIALISASI LINKED LIST (DOUBLE LINKED LIST)

- Operasi inisialisasi double linked list sangat mirip dengan inisialisasi di single linked list, tetapi diperlukan tambahkan operasi.
- Operasi tambahannya adalah adanya inisialisasi untuk akhir dari linked list.

INISIALISASI LINKED LIST (DOUBLE LINKED LIST)

- Ilustrasi inisialiasi linked list

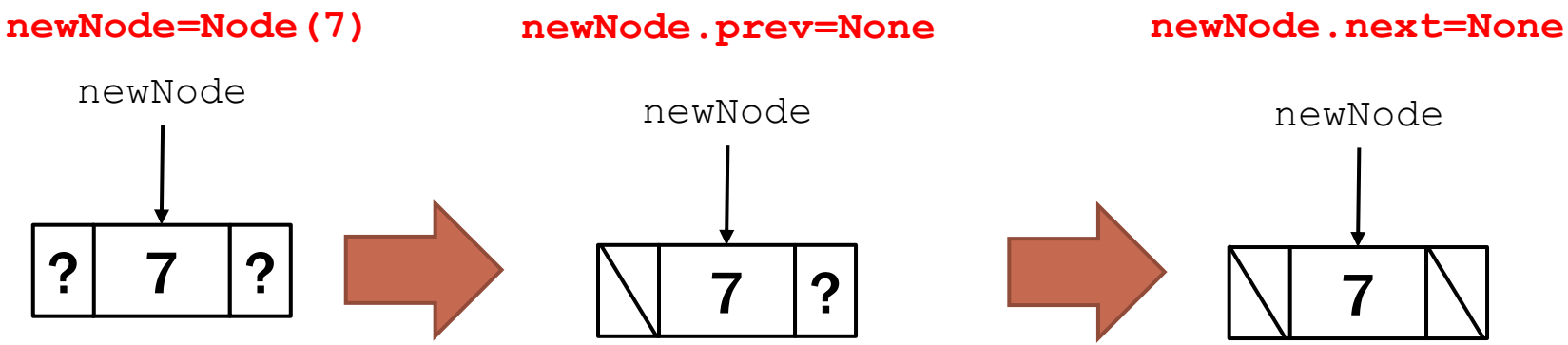


MEMBUAT NODE BARU (DOUBLE LINKED LIST)

- Langkah-langkah membuat node baru pada double linked list sangat mirip dengan pembuatan node pada single linked list.
- Perbedaannya adalah adanya inisialisasi dengan nilai NIL atau NULL pada field **`prev`**.

MEMBUAT NODE BARU (DOUBLE LINKED LIST)

- Ilustrasi pembuatan node baru.
  - Contoh : buat node baru dengan data berisi nilai 7,



PEMERIKSAAN KONDISI KOSONG LINKED LIST

- Proses pemeriksaan kondisi kosong pada double linked list **sama saja dengan** pemeriksaan kondisi kosong pada single linked list.
- Sebuah double linked list disebut kosong jika pointer yang menunjuk ke awal data bernilai NIL/NULL/None.
- Anda dapat mengimplementasikan operasi ini dalam method bernama **`isEmpty`**

TRAVERSAL DI DOUBLE LINKED LIST

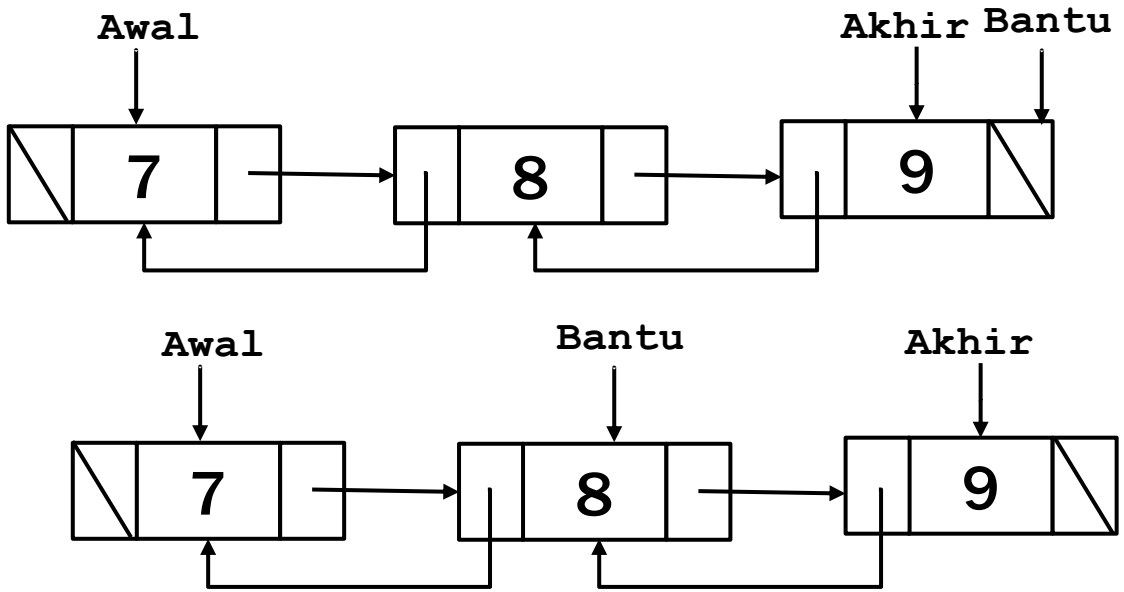
- Traversal dalam double linked list bisa 2 arah yaitu :
  - Traversal maju yaitu menelusuri linked list dari pointer awal menuju akhir (dengan memanfaatkan pointer `next` pada node)
  - Traversal mundur yaitu menelusuri linked list dari pointer akhir menuju awal (dengan menanfaatkan pointer `prev` pada node).
- Langkah-langkah operasi traversal maju dalam double linked list sama saja dengan operasi traversal dalam single linked list.
- Operasi traversal untuk menampilkan data dari awal linked list menuju akhir linked list pada double linked list dapat diimplementasikan dalam function/method `display()` yang source-codenya sama dengan method display pada single linked list.
- Sedangkan operasi traversal untuk menampilkan data dari akhir linked list menuju awal linked list pada double linked list memiliki perbedaan karena pointer yang digunakan adalah pointer `prev` pada node. Untuk operasi ini, anda dapat mengimplementasikannya dalam sebuah node `displayReverse()`.

TRAVERSAL DI DOUBLE LINKED LIST

- Langkah-langkah untuk menampilkan data dalam linked list dari akhir menuju awal adalah :
  1. Periksa apakah linked listnya kosong
  2. Jika linked list dalam kondisi kosong, maka tampilkan pesan bahwa “Data Kosong”.
  3. Jika linked list dalam kondisi tidak kosong, maka lakukan traversal linked list untuk menampilkan data.
     A. Buat sebuah pointer yang akan digunakan untuk menelusuri linked list (sebut sebagai variable **`Bantu`**).
     B. **`Bantu`** dimulai dari akhir linked list.
     C. Ulangi langkah D dan E selama variable **`Bantu`** tidak nil (ujung awal linked list)
     D. Tampilkan data (info) pada node ditunjuk oleh **`Bantu`**.
     E. Pindahkan **`Bantu`** ke node sebelumnya (**`Bantu.prev`**).

TRAVERSAL DI DOUBLE LINKED LIST

- Ilustrasi traversal terbalik (**displayReverse()**) adalah :

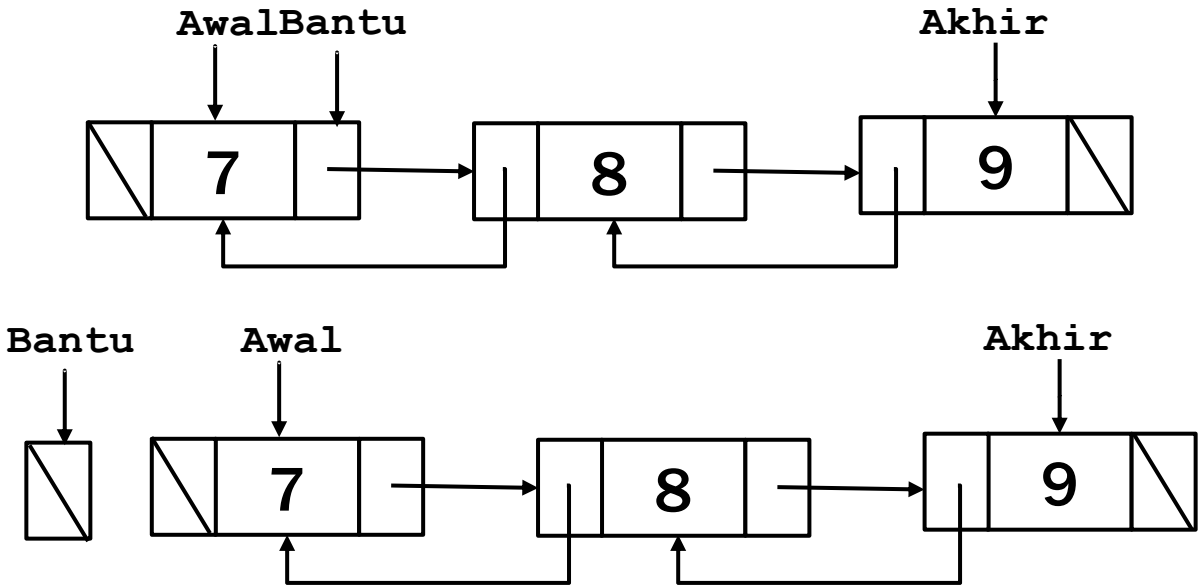



```
Bantu = getLast()

Print(Bantu.info) # Menampilkan 9
```

```
Bantu = Bantu.prev

Print(Bantu.info) # Menampilkan 8
```

TRAVERSAL DI DOUBLE LINKED LIST

- Ilustrasi traversal terbalik (**displayReverse()**) adalah :




```
Bantu = Bantu.prev

Print(Bantu.info) # Menampilkan 7
```

```
Bantu = Bantu.prev

Pengulangan berhenti karena bantu
menunjuk ke NIL/None
```

PENCARIAN NODE BERDASAR POSISI

- Proses pencarian node digunakan untuk mendapatkan alamat dari sebuah node pada posisi tertentu di dalam linked list.
- Proses ini dibuat untuk mempermudah proses pencarian node ketika akan melakukan proses seperti penambahan di akhir, penambahan di awal, penghapusan di tengah dan lain-lain.
- Ada tiga pencarian node yang bisa dilakukan
  - Mencari node yang berada di awal linked list (`getFirst`)
  - Mencari node yang berada di akhir linked list (`getLast`).
  - Mencari node yang berada di posisi tertentu di dalam linked list (`get`)

PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AWAL LINKED LIST)

- Pencarian node di awal linked list digunakan untuk mendapatkan pointer yang menunjuk ke awal linked list (tentu saja).
- Proses ini sebenarnya sangat sederhana yaitu cukup dengan mereturnkan nilai awal linked list saja. Tetapi untuk mempermudah keterbacaan source code, maka sebaiknya dibuat dalam bentuk function.
- Function pencarian node awal linked list (**getFirst**)

```
def getFirst(self):
        return self.awal
```

PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI AKHIR DOUBLE LINKED LIST)

- Pencarian node di akhir linked list digunakan untuk mendapatkan pointer yang menunjuk ke node di akhir double linked list.
- Dikarenakan double linked list yang digunakan memiliki node Akhir, maka anda cukup membuat sebuah function yang mereturnkan nilai yang dipegang oleh Akhir dari linked list.
- Function pencarian node akhir linked list (**getLast**)

```
def getLast(self):
        return self.akhir
```

PENCARIAN NODE BERDASAR POSISI (PENCARIAN NODE DI POSISI TERTENTU LINKED LIST)

- Pencarian node di posisi tertentu linked list digunakan untuk mendapatkan pointer yang menunjuk ke node yang berada pada posisi tertentu di dalam double linked list.
- Algoritma pencarian node di posisi tertentu pada double linked list sebenarnya sama saja dengan pencarian node di posisi tertnetu pada single linked list.
- Anda dapat meng-copy-paste method `get` dari class LinkedList ke class DoubleLinkedList.

PENAMBAHAN DATA DI AWAL (DOUBLE LINKED LIST)

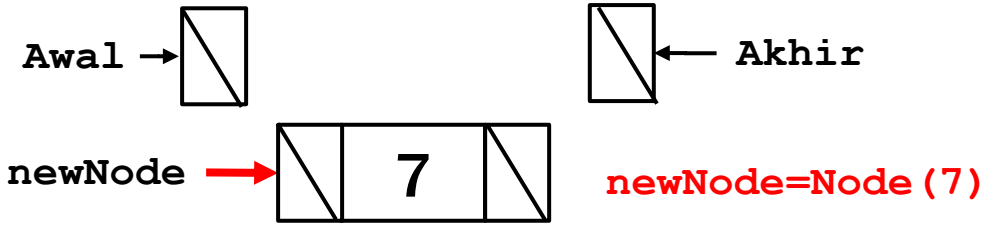
- Penambahan data di awal pada double linked list mirip dengan penambahan data di awal pada single linked list, tetapi ada langkah-langkah tambahan.
- Langkah-langkah tambahannya adalah untuk mengelola link dari link yang menyambungkan field **`prev`** dari tiap-tiap node.

PENAMBAHAN DATA DI AWAL (DOUBLE LINKED LIST)

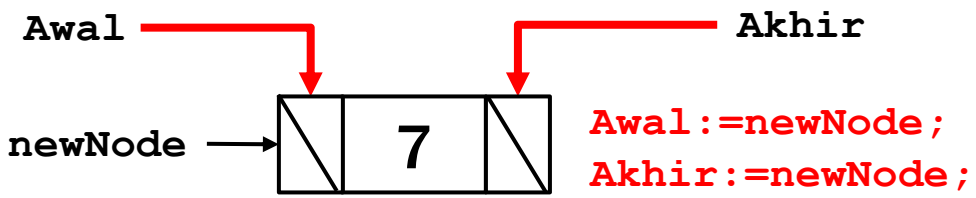
- Ilustrasi penambahan data di awal
  - Keadaan linked list masih kosong

Kondisi awal

Buat node baru (newNode)



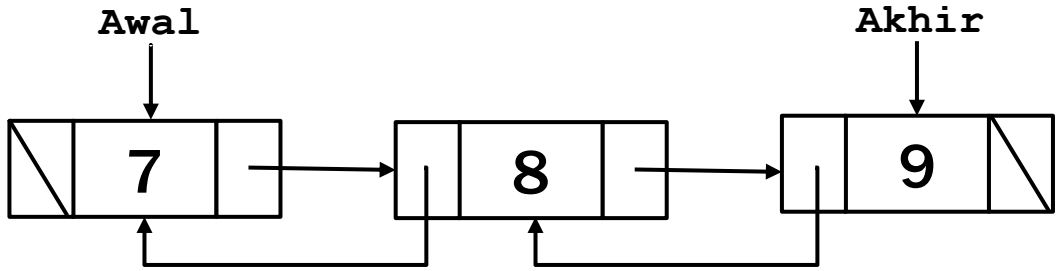
Sambungkan Awal dan Akhir ke node baru



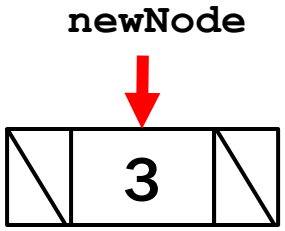
PENAMBAHAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di awal
  - Keadaan linked list sudah ada data

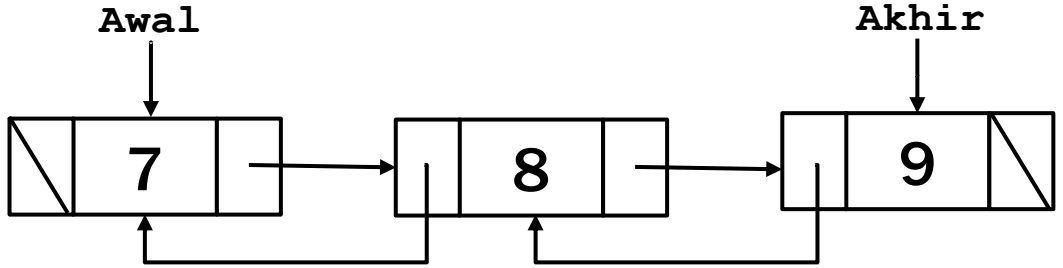
Kondisi awal



1. Buat node baru (newNode)



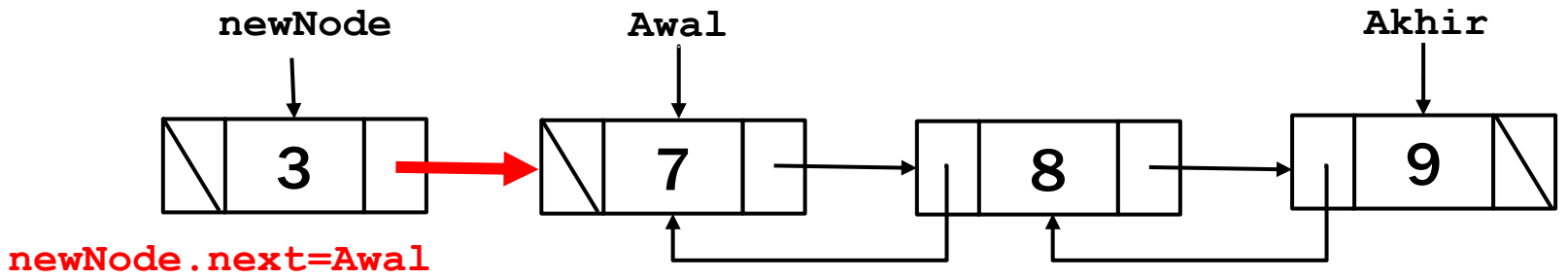
newNode=Node(3)



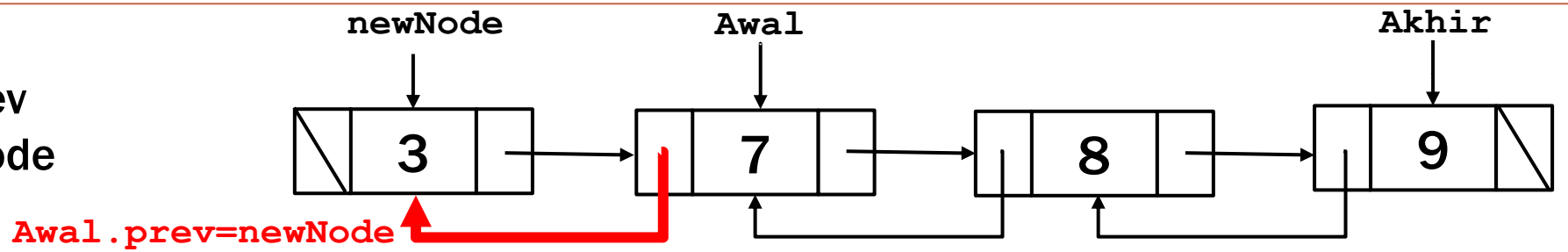
PENAMBAHAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di awal
  - Keadaan linked list sudah ada data

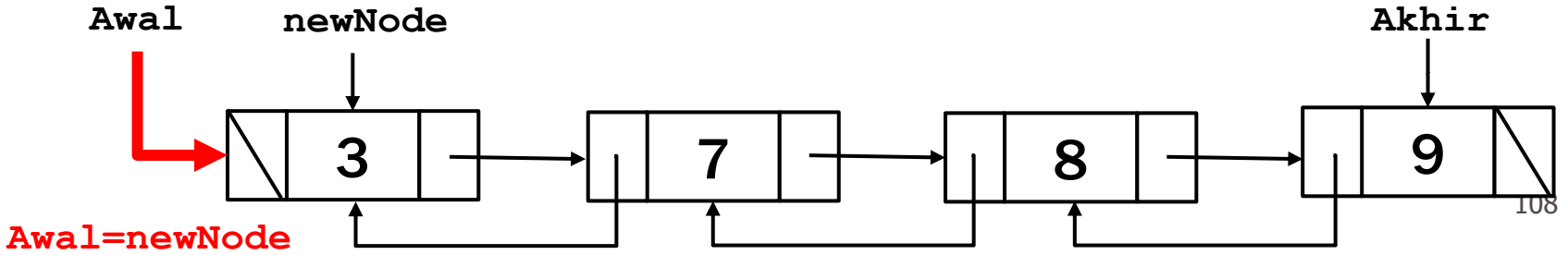
2. Sambungkan next dari newNode ke Awal



3. Sambungkan prev dari Awal ke newNode



4. Pindahkan Awal ke newNode



PENAMBAHAN DATA DI AKHIR (DOUBLE LINKED LIST)

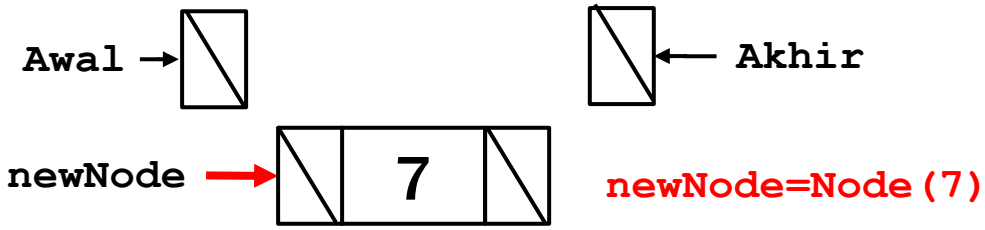
- Penambahan data di akhir pada double linked list mirip dengan penambahan data di akhir pada single linked list, tetapi ada perubahan langkah, yaitu :
  - Karena ada node yang menunjuk ke akhir, maka tidak perlu lagi melakukan pencarian node terakhir.
  - Setelah menyambungkan node terakhir di akhir linked list, jangan lupa untuk menghubungkan prev dari node terakhir ke node terakhir sebelumnya.

PENAMBAHAN DATA DI AKHIR (DOUBLE LINKED LIST)

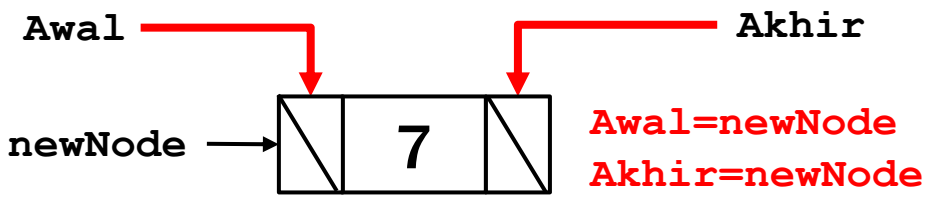
- Ilustrasi penambahan data di akhir
  - Keadaan linked list masih kosong (sama saja dengan tambah di awal)

Kondisi awal

Buat node baru (newNode)



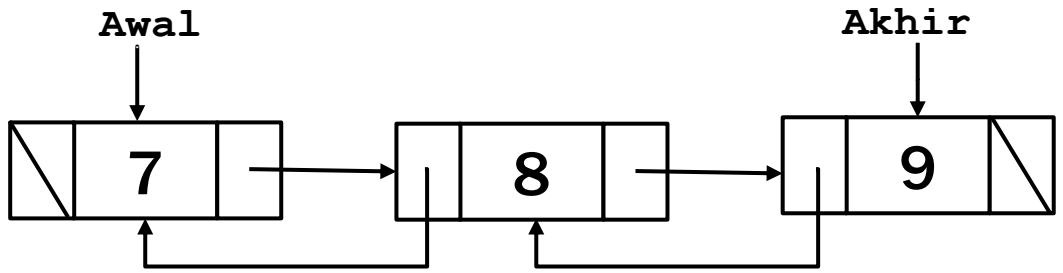
Sambungkan Awal dan Akhir ke node baru



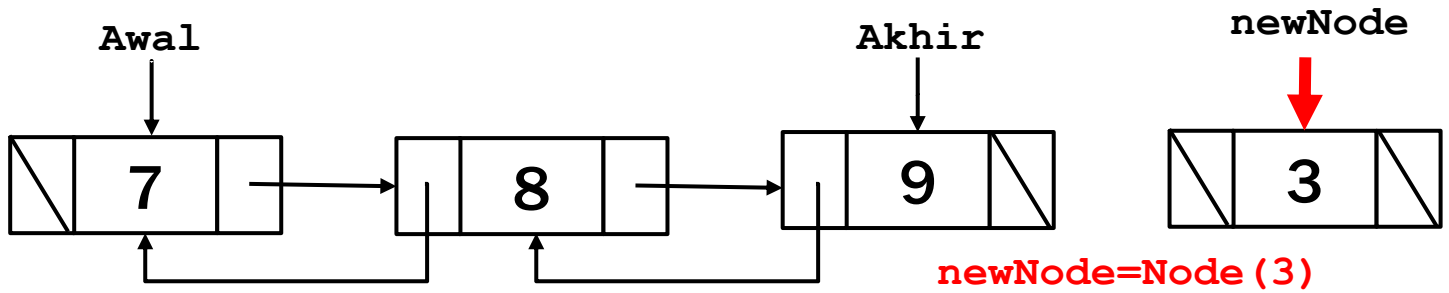
PENAMBAHAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di akhir
  - Keadaan linked list sudah ada data

Kondisi awal



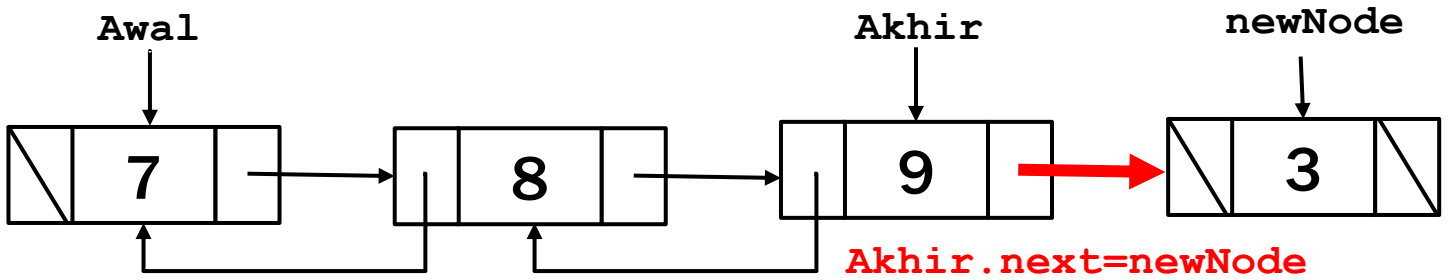
1. Buat node baru (newNode)



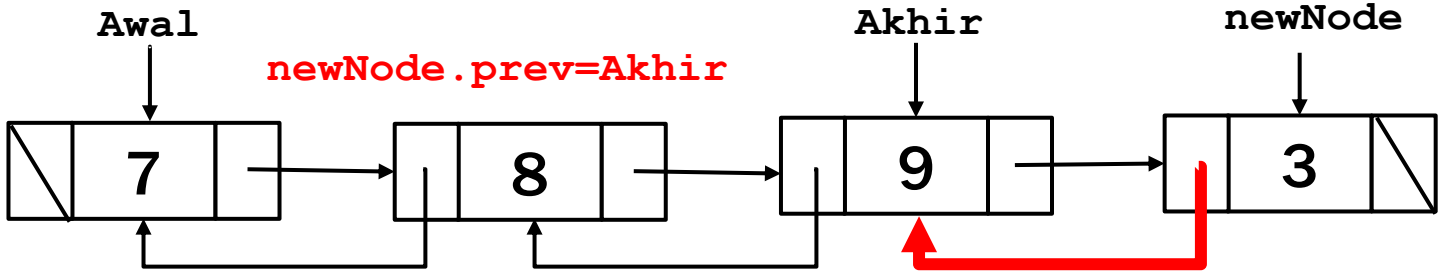
PENAMBAHAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di akhir
  - Keadaan linked list sudah ada data

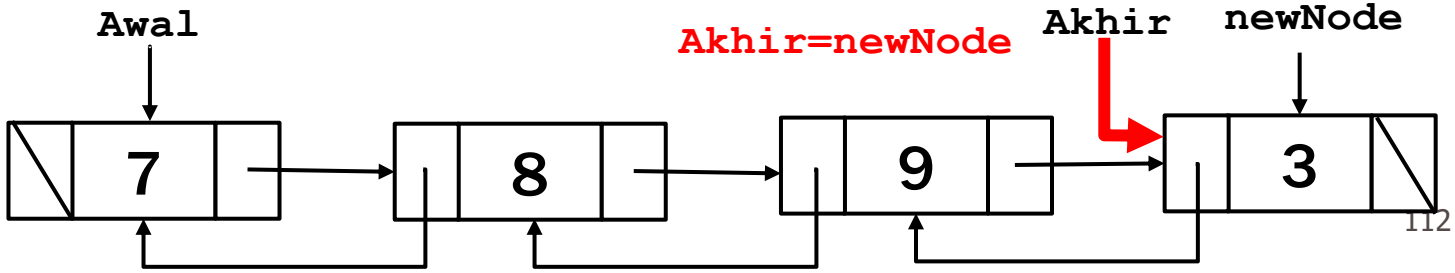
2. Sambungkan next dari Akhir ke newNode



3. Sambungkan prev dari newNode ke Akhir



4. Pindahkan Akhir ke newNode



PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

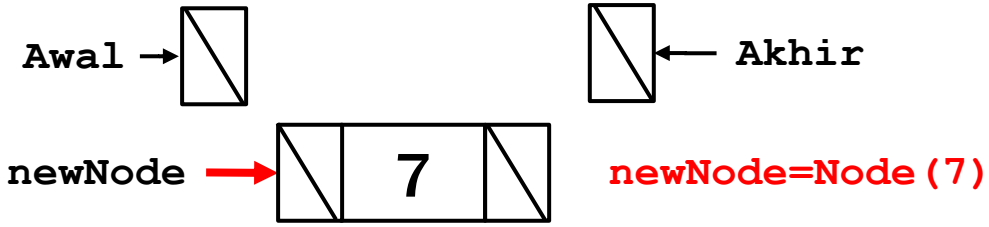
- Penambahan data di posisi tertentu pada double linked list mirip dengan penambahan data di posisi tertentu pada single linked list, tetapi ada perubahan langkah, yaitu :
  - Untuk menyambungkan node baru tidak harus mencari node sebelum posisi penyisipan, karena node sebelum bisa diketahui dari field prev dari node yang berada di posisi penyisipan.
  - Jangan lupa sambungkan node prev dari node baru ke node yang berada di posisi sebelum node sisip.
  - Sambungkan juga next dari node baru ke node yang berada di posisi penyisipan.

PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

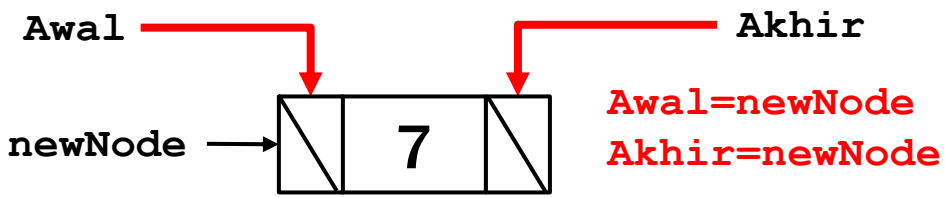
- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list masih kosong (sama saja dengan tambah di awal)

Kondisi awal

Buat node baru (newNode)



Sambungkan Awal dan Akhir ke node baru



PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

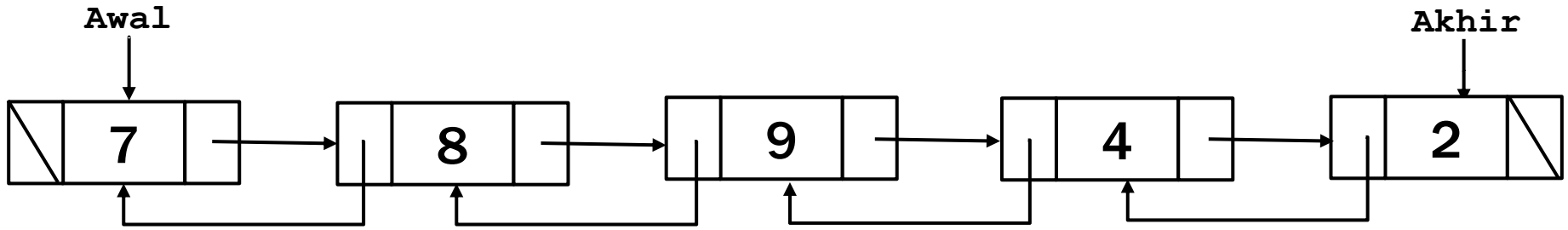
- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list sudah ada data dan posisi penyisipan ada di index ke-1

Sama Dengan Tambah Di Awal

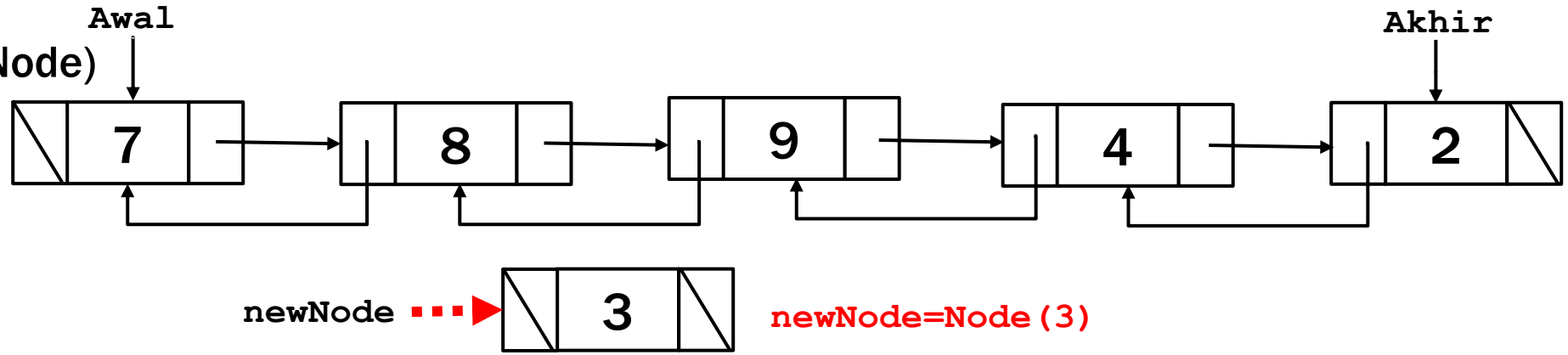
PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list sudah ada data dan penyisipan dilakukan pada index 3.

Kondisi awal



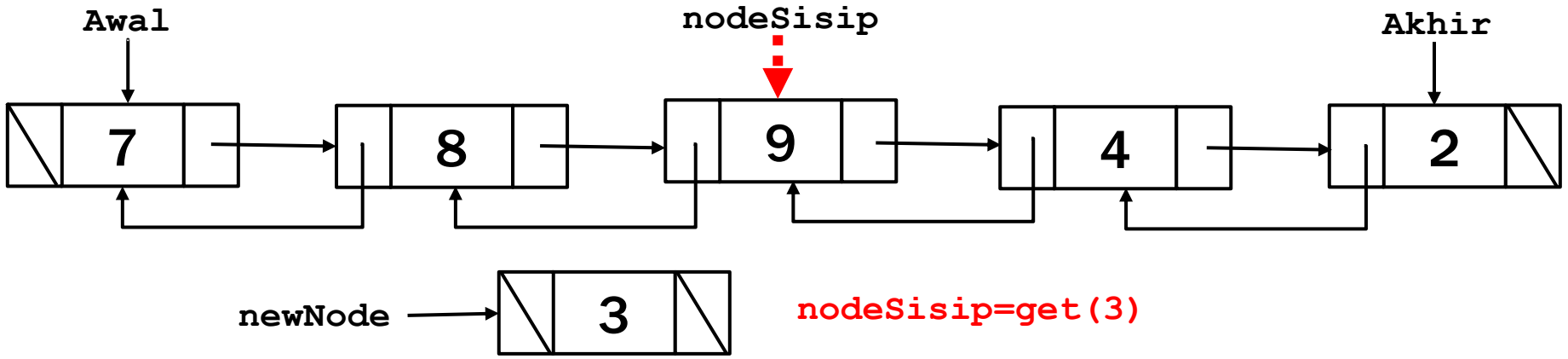
1. Buat node baru (diberi nama newNode)



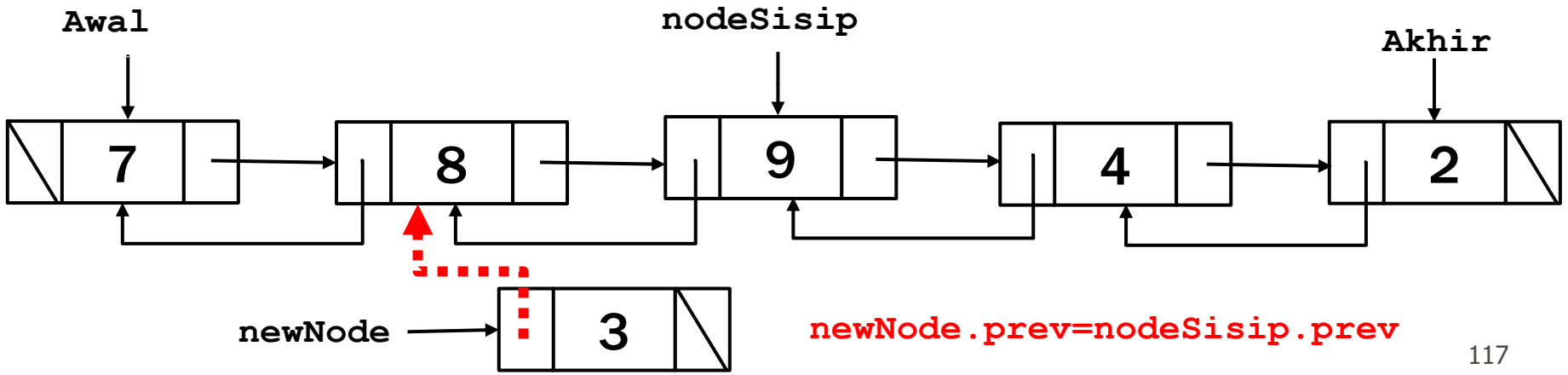
PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list sudah ada data dan penyisipan dilakukan pada index 3.

2. Cari node yang menunjuk di posisi sisip (disebut nodeSisip)



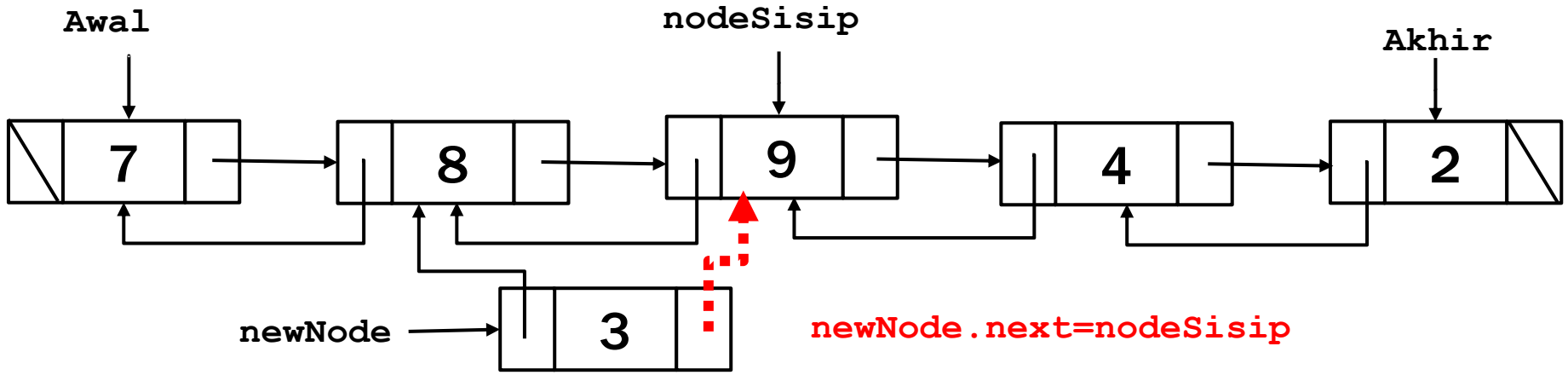
3. Sambungkan prev dari newNode ke node yang ditunjuk prev dari nodeSisip



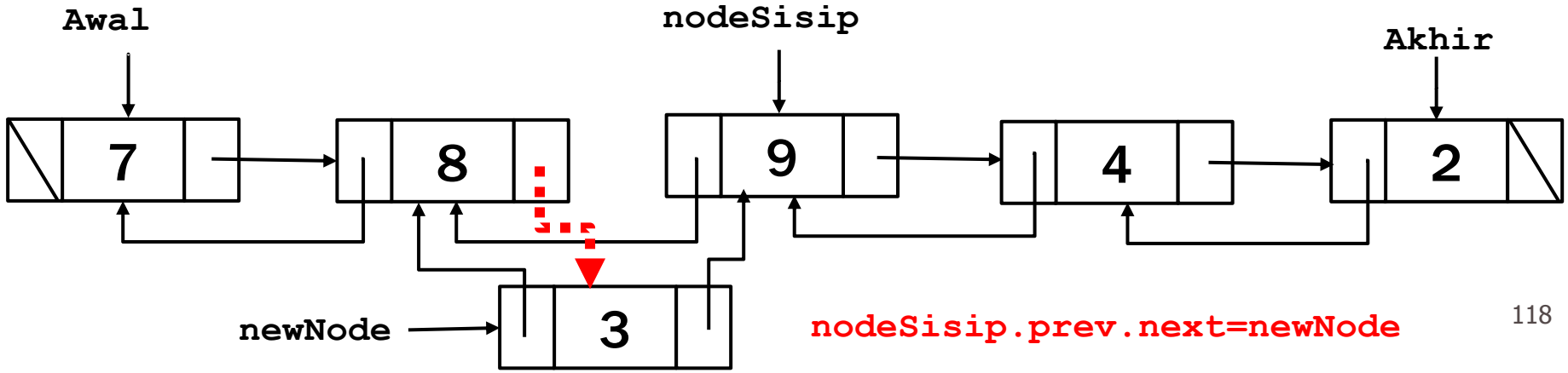
PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list sudah ada data dan penyisipan dilakukan pada index 3.

4. Sambungkan next dari newNode ke nodeSisip



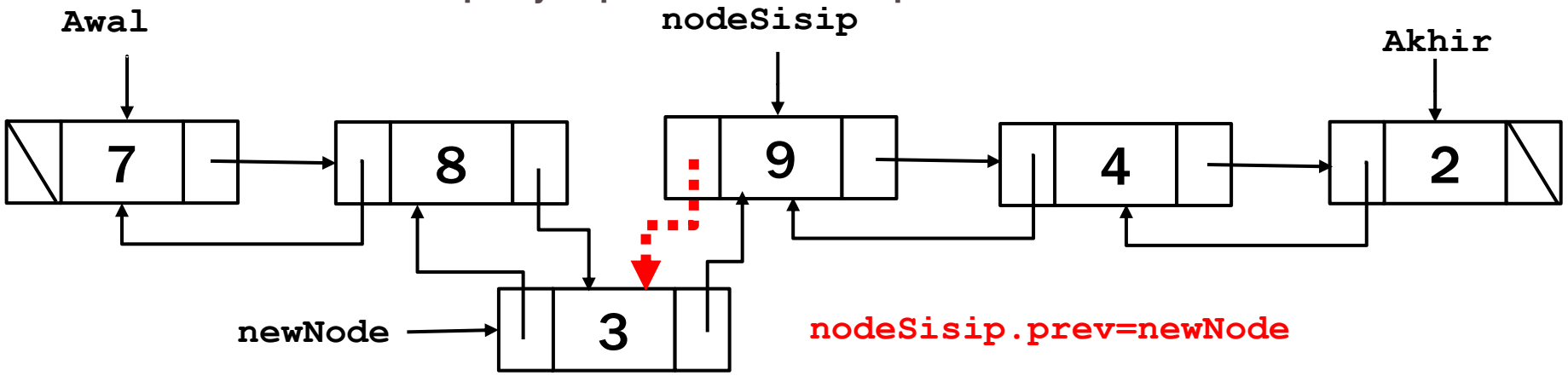
5. Sambungkan next dari node sebelum nodeSisip ke newNode



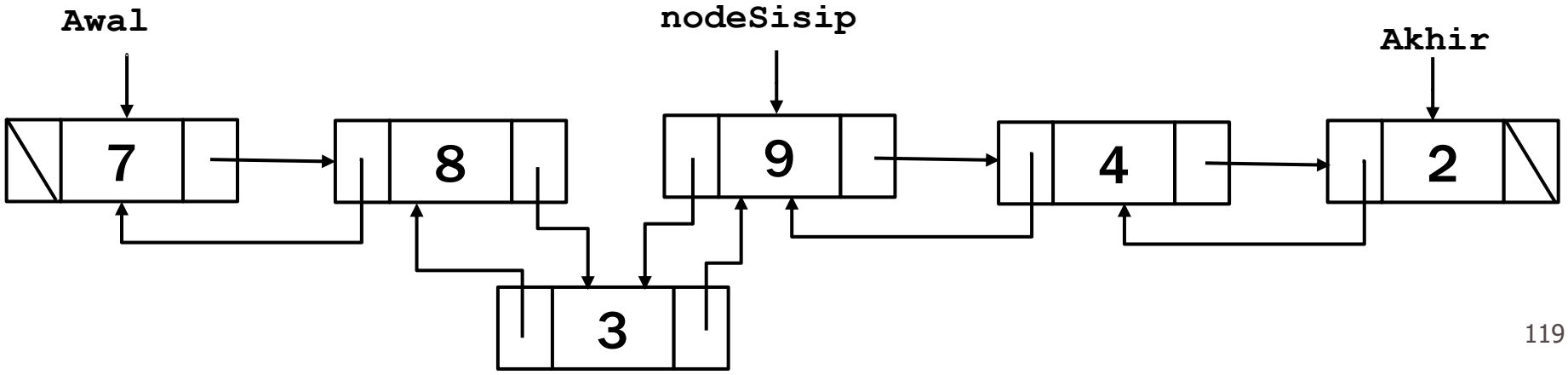
PENAMBAHAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penambahan data di posisi tertentu
  - Keadaan linked list sudah ada data dan penyisipan dilakukan pada index 3.

6. Sambungkan prev dari nodeSisip ke newNode



Kondisi Akhir



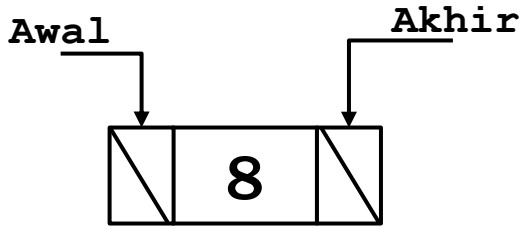
PENGHAPUSAN DATA (DOUBLE LINKED LIST)

- Penghapusan data dalam double linked list memiliki langkah-langkah seperti penghapusan data dalam single linked list.
- Perbedaannya hanyalah adanya langkah-langkah tambahan untuk menangani link yang menunjuk ke link sebelumnya (`prev`).
- Catatan:
  - Ada langkah yang berbeda ketika akan menghapus node dalam kondisi banyak node hanya satu dengan dalam kondisi banyak node lebih dari 1.
  - Ketika anda melakukan penghapusan ketika banyak node di dalam linked list hanya 1 maka nilai penunjuk Awal dan Akhir node akan berubah, tetapi hal tersebut tidak berlaku ketika banyak node dalam linked list lebih dari 1.

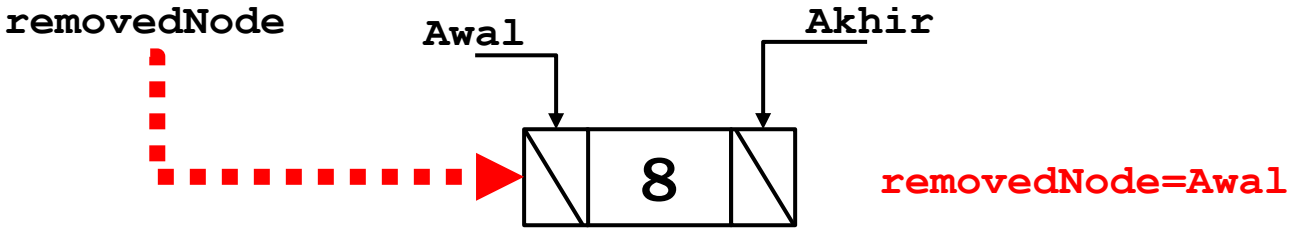
PENGHAPUSAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node awal (jika banyak data/node dalam linked list = 1)

Kondisi awal



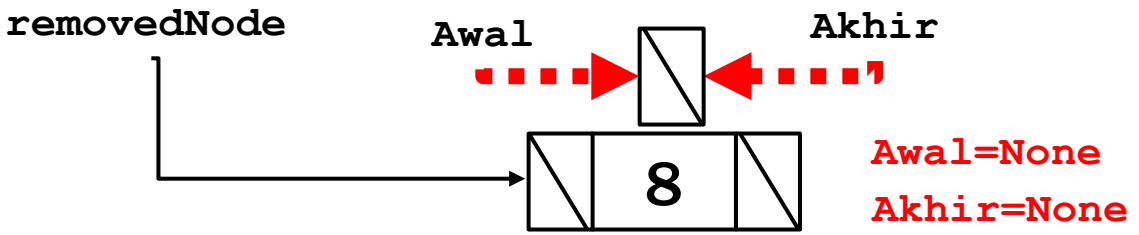
1. Simpan node yang menunjuk awal linked list ke removedNode



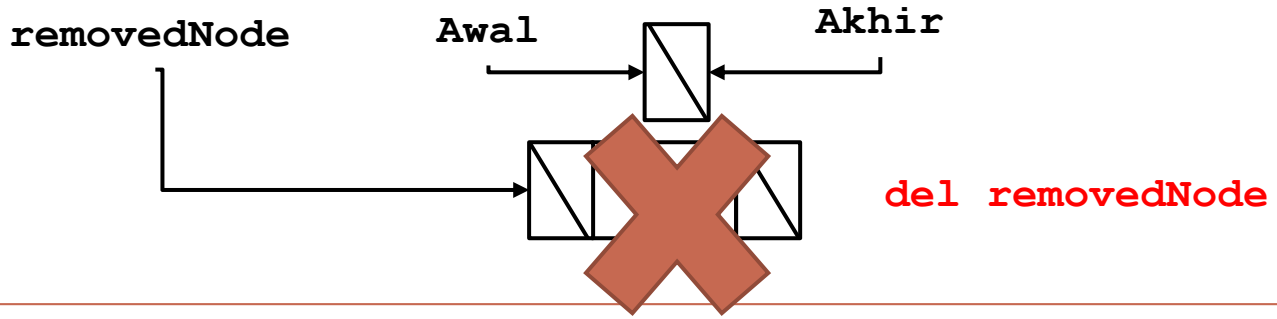
PENGHAPUSAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node awal (jika banyak data/node dalam linked list = 1)

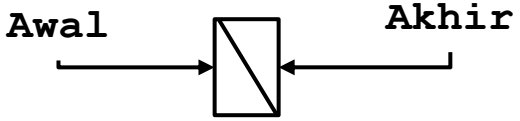
2. NIL-kan node Awal dan Akhir



3. Hapus node yang ditunjuk oleh removedNode



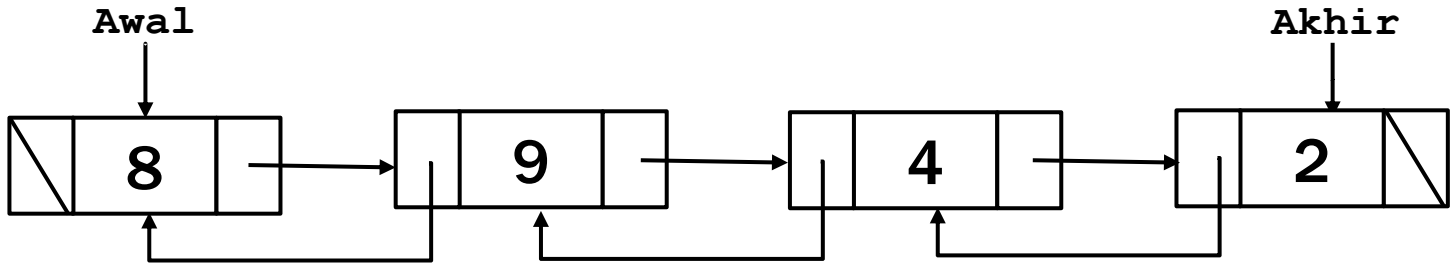
Kondisi Akhir



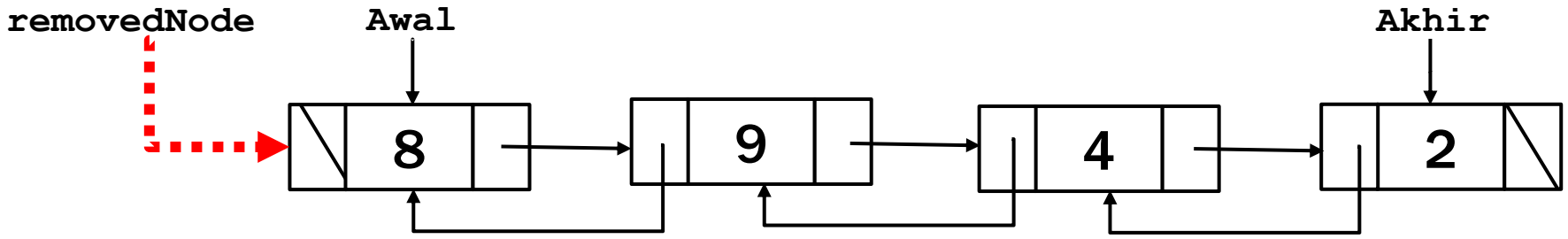
PENGHAPUSAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node awal ((jika banyak data/node dalam linked list > 1)

Kondisi awal



1. Simpan node yang menunjuk awal linked list ke removedNode

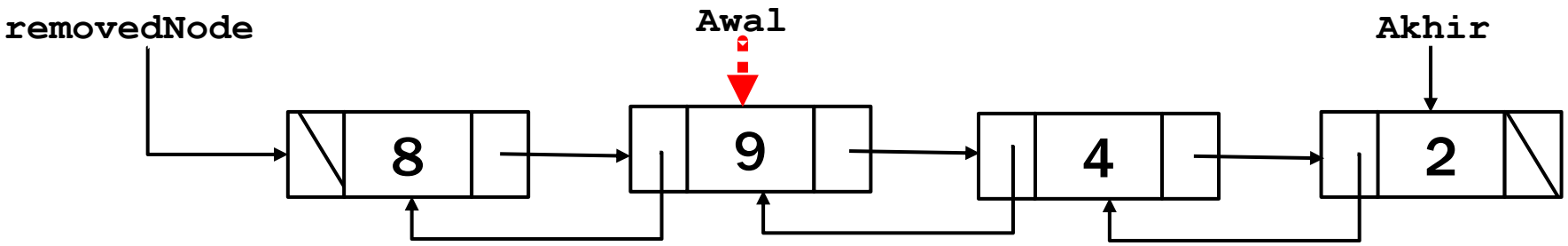


removedNode=Awal

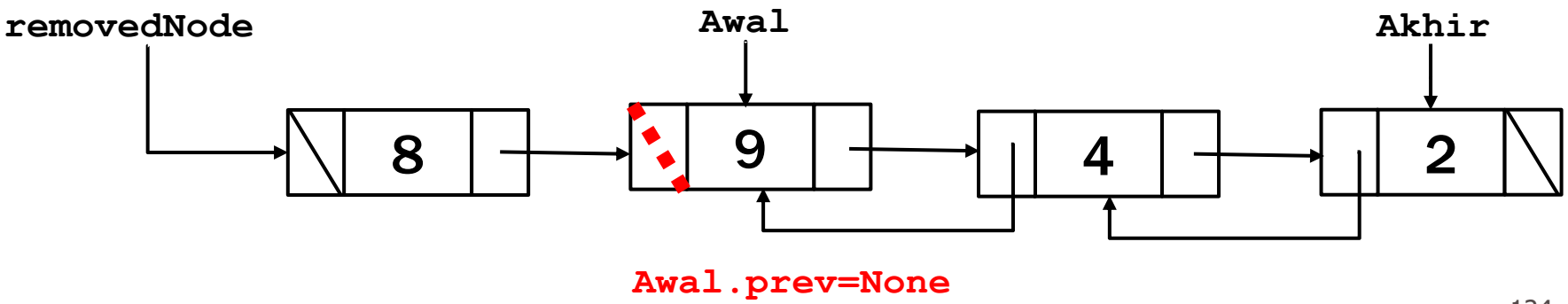
PENGHAPUSAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node awal (jika banyak data/node dalam linked list > 1)

2. Pindahkan node Awal ke node berikutnya.



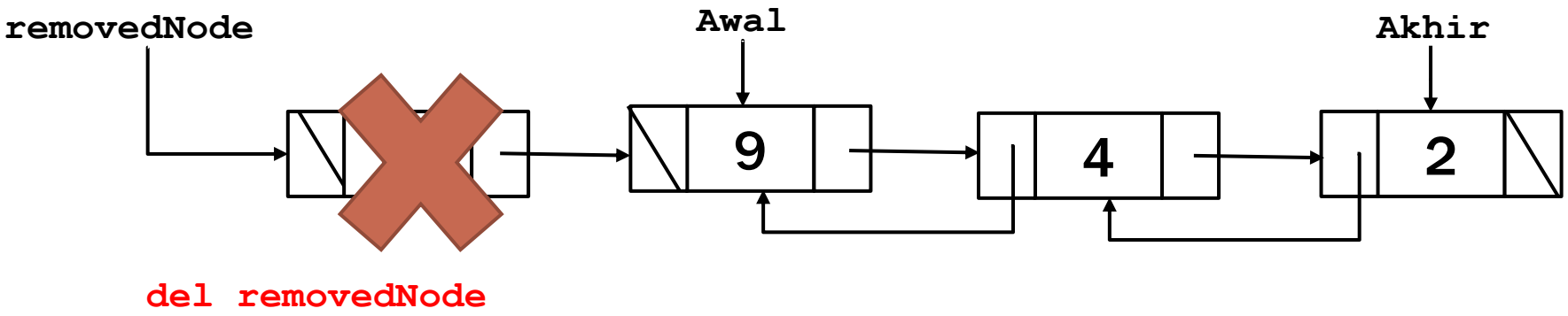
3. Link prev dari node Awal di-NIL-kan



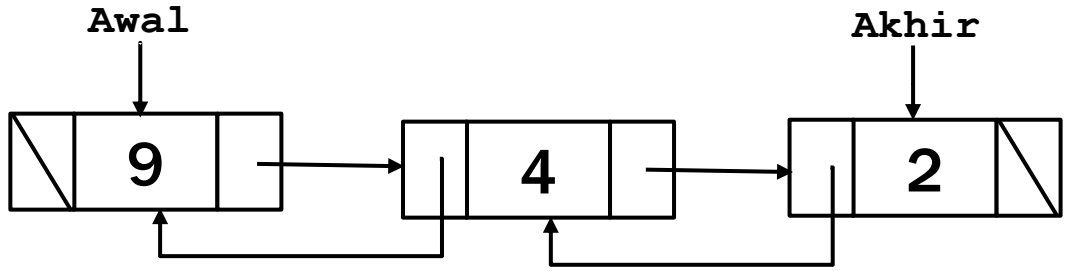
PENGHAPUSAN DATA DI AWAL (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node awal ((jika banyak data/node dalam linked list > 1)

4. Hapus node removedNode



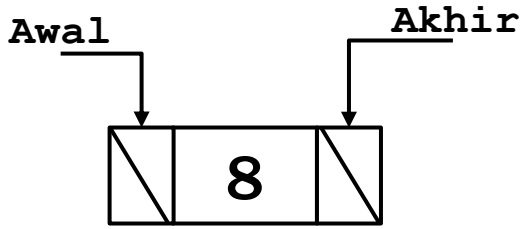
Kondisi Akhir



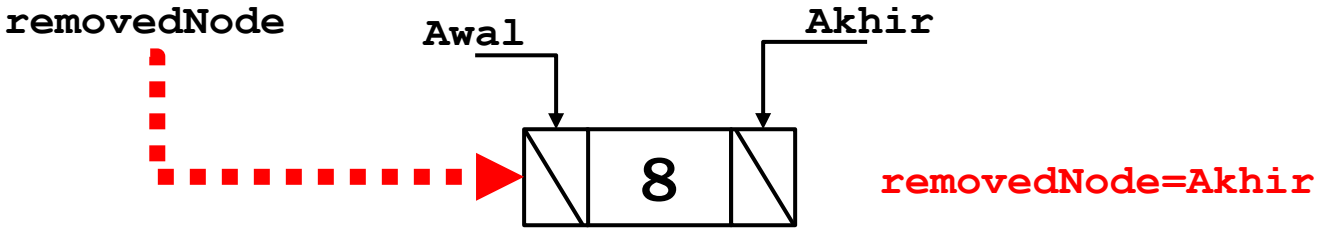
PENGHAPUSAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node akhir (jika banyak data/node dalam linked list = 1)

Kondisi awal



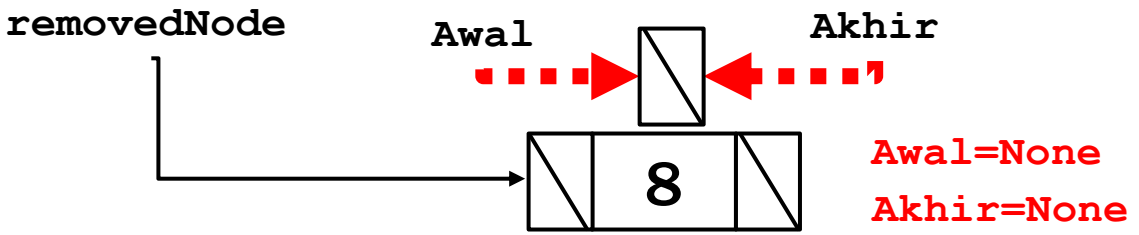
1. Simpan node yang menunjuk Akhir linked list ke removedNode



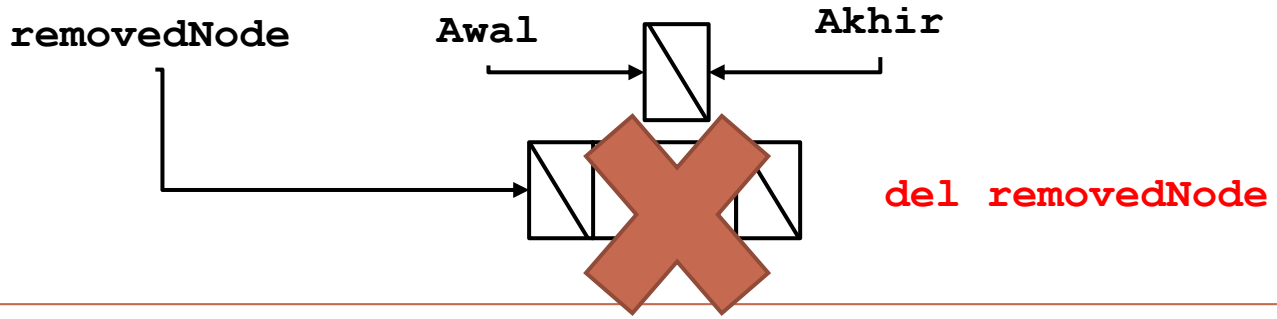
PENGHAPUSAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node akhir (jika banyak data/node dalam linked list = 1)

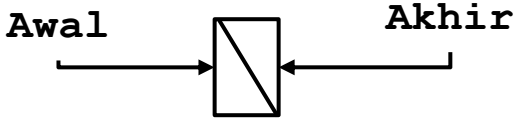
2. NIL-kan node Awal dan Akhir



3. Hapus node yang ditunjuk oleh removedNode



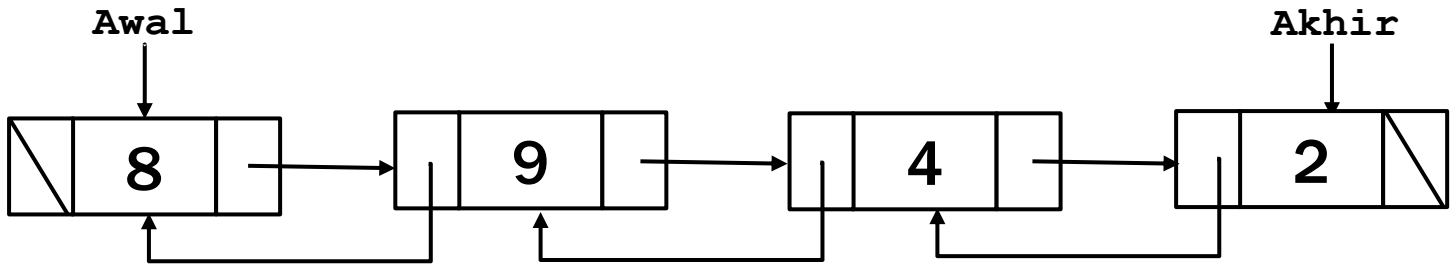
Kondisi Akhir



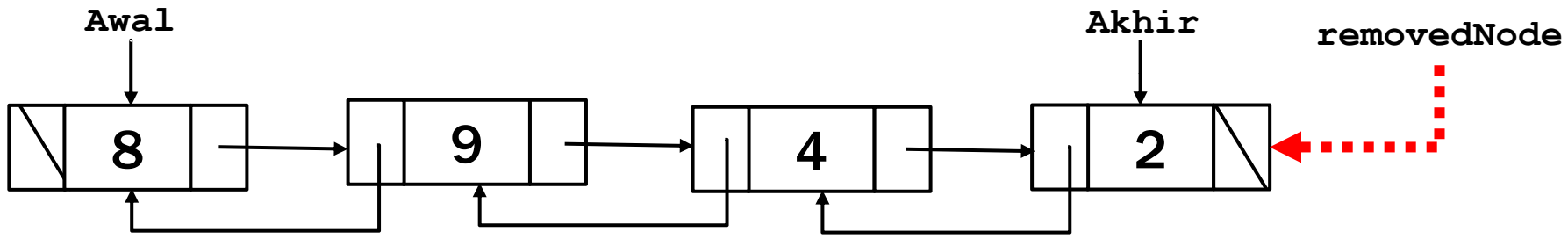
PENGHAPUSAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node akhir (jika banyak data/node dalam linked list > 1)

Kondisi awal



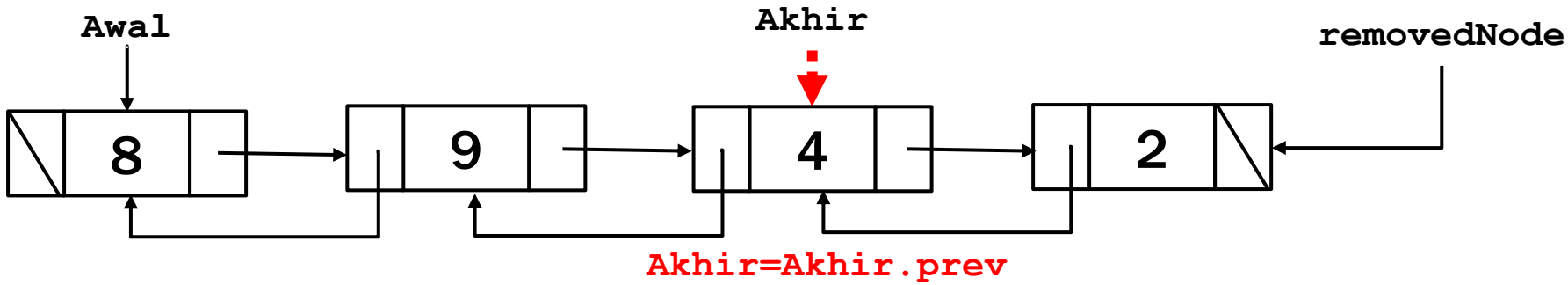
1. Simpan node yang menunjuk Akhir linked list ke removedNode



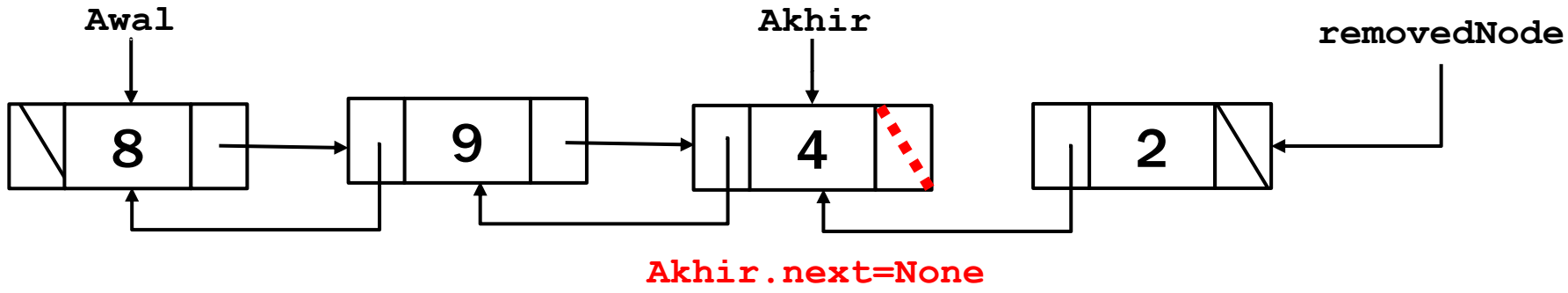
PENGHAPUSAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node akhir (jika banyak data/node dalam linked list > 1)

2. Pindahkan node Akhir ke node sebelumnya



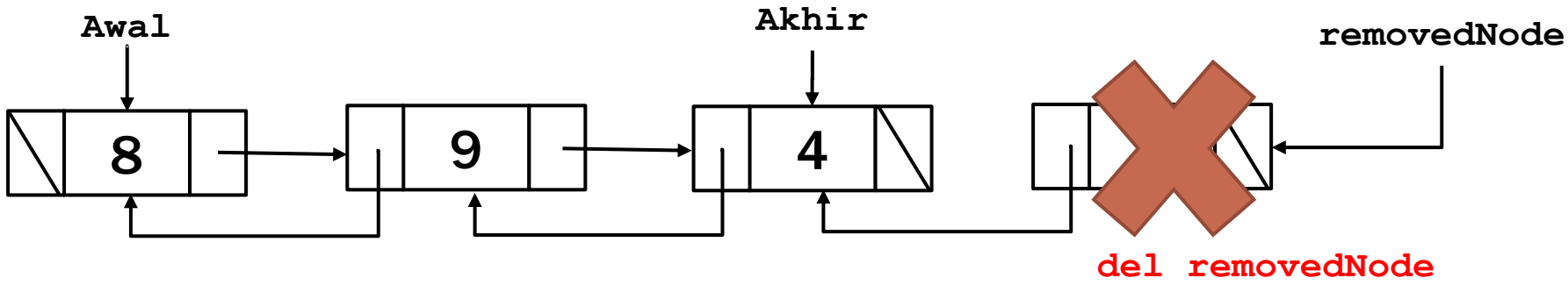
3. Ubah next dari Akhir ke nilai NIL.



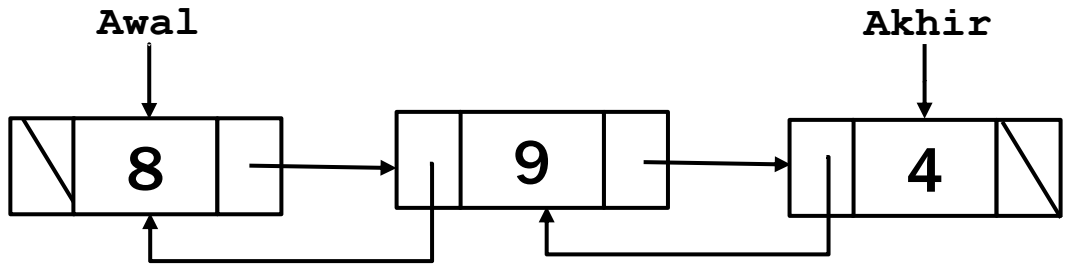
PENGHAPUSAN DATA DI AKHIR (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node akhir (jika banyak data/node dalam linked list > 1)

4. Hapus node removedNode



Kondisi Akhir



PENGHAPUSAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node tertentu
  - Jika index node yang akan dihapus sama dengan 1

Sama Dengan Hapus Di Awal

```
removeFirst()
```

PENGHAPUSAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node tertentu
  - Jika index node yang akan dihapus sama dengan index terakhir.

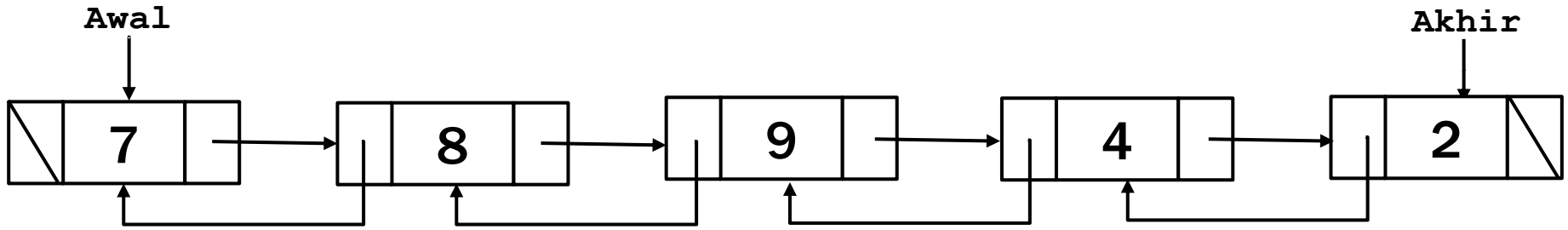
Sama Dengan Hapus Di Akhir

`removeLast()`

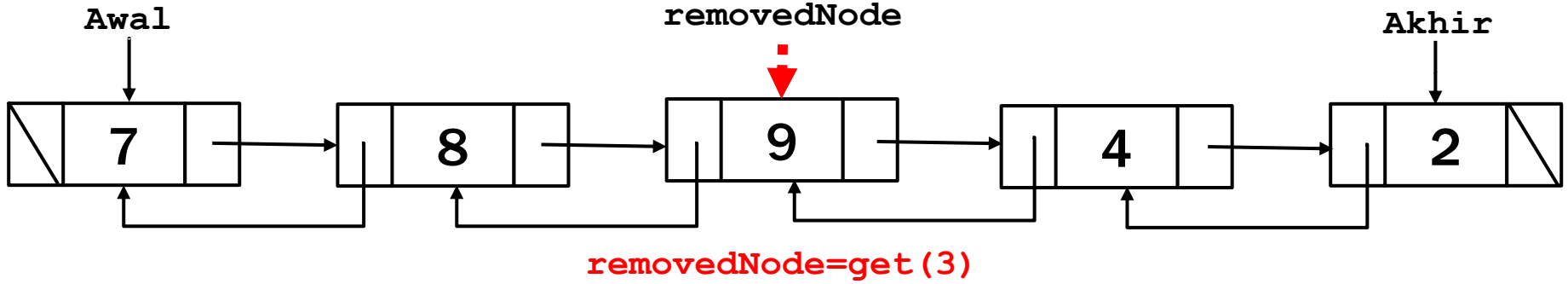
PENGHAPUSAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node tertentu
  - Jika index node yang akan dihapus berada di antara node awal dan node akhir. Contoh penghapusan node di index ke-3

Kondisi awal



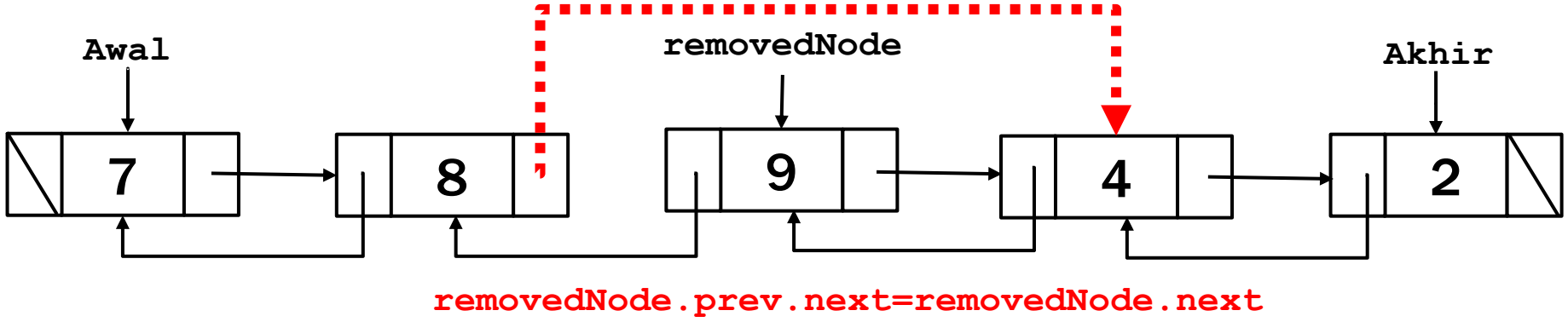
1. Cari node di posisi hapus, diberi nama removedNode



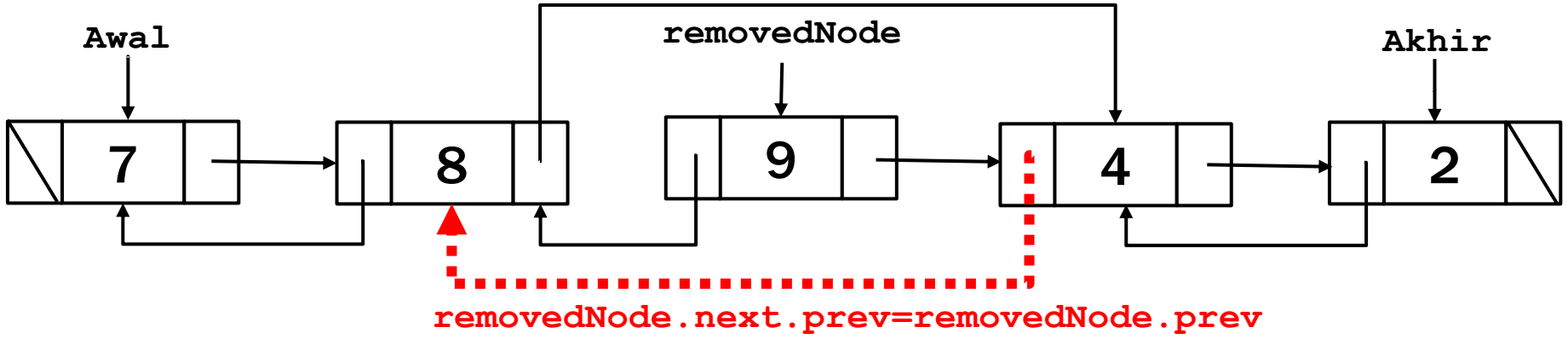
PENGHAPUSAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node tertentu
  - Jika index node yang akan dihapus berada di antara node awal dan node akhir. Contoh penghapusan node di index ke-3

2. Sambungkan node next dari node sebelum removeNode ke node setelah removedNode



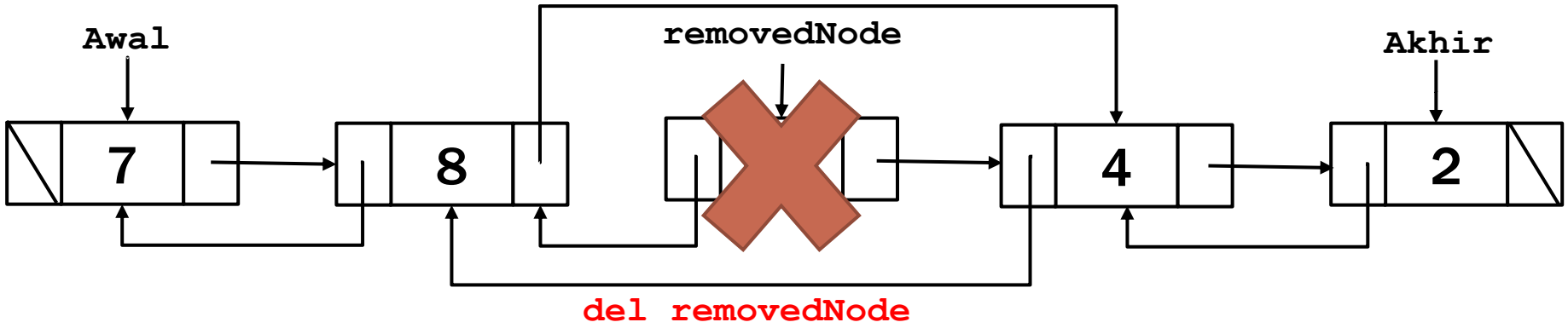
3. Sambungkan node prev dari node setelah removeNode ke node sebelum removedNode



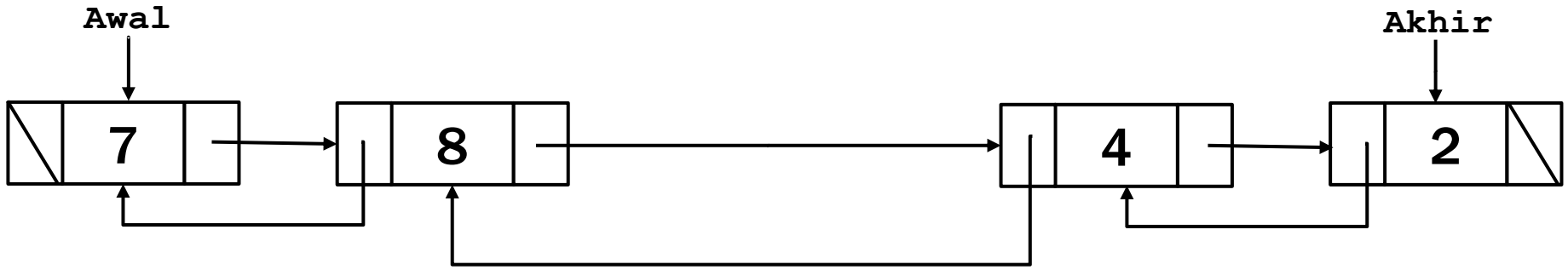
PENGHAPUSAN DATA DI POSISI TERTENTU (DOUBLE LINKED LIST)

- Ilustrasi penghapusan data di node tertentu
  - Jika index node yang akan dihapus berada di antara node awal dan node akhir. Contoh penghapusan node di index ke-3

4. Hapus node removedNode



Kondisi Akhir



FORUM DISKUSI

LMS UNIKOM
https://lms.unikom.ac.id

Group Whatsapp Perkuliahan

Youtube Comments