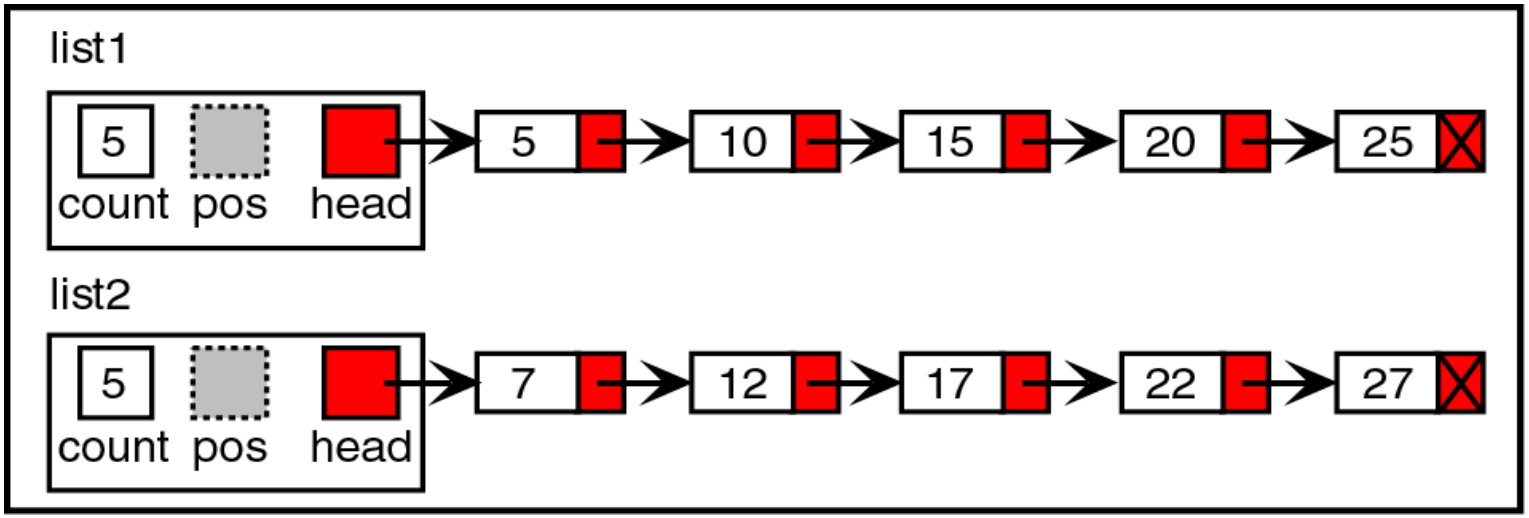
BLM212 Veri Yapıları

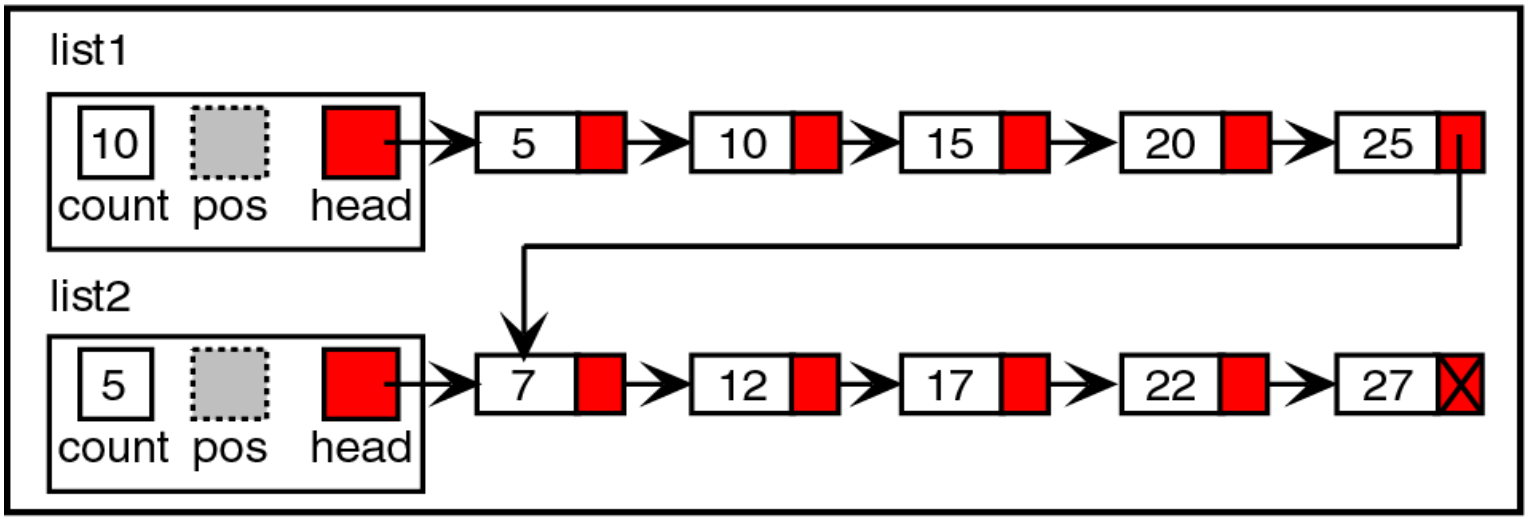
Linear Lists – part2 (Doğrusal Listeler)

2021-2022 Güz Dönemi

Linear List Applications
Array of Linked Lists



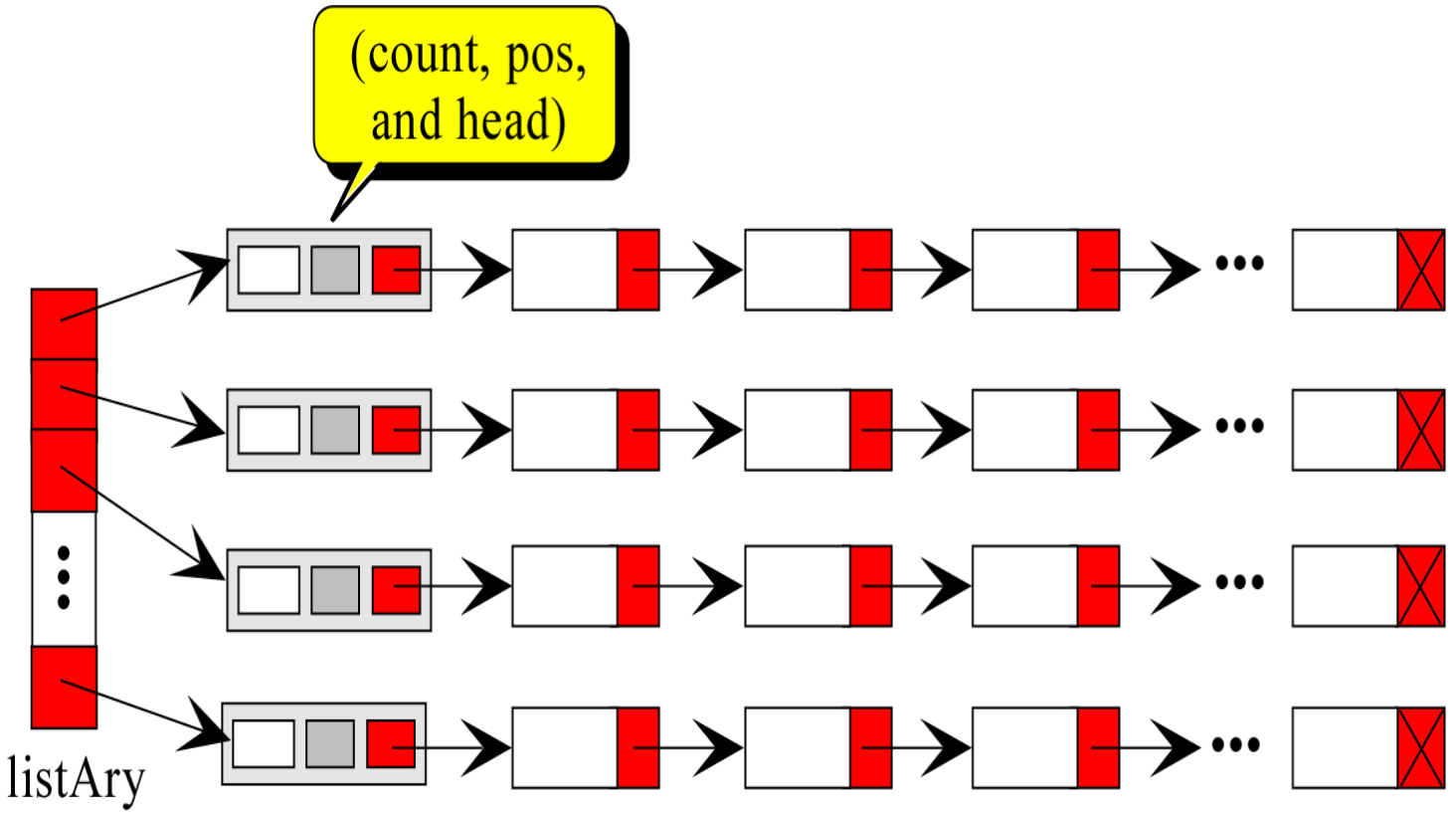
(a) Before Append



(b) After Append

Linear List Applications
Array of Linked Lists

- Her bir bağlı liste dizideki bir satırı temsil eder.
- Bağlı listedeki düğümler sütunları temsil eder.



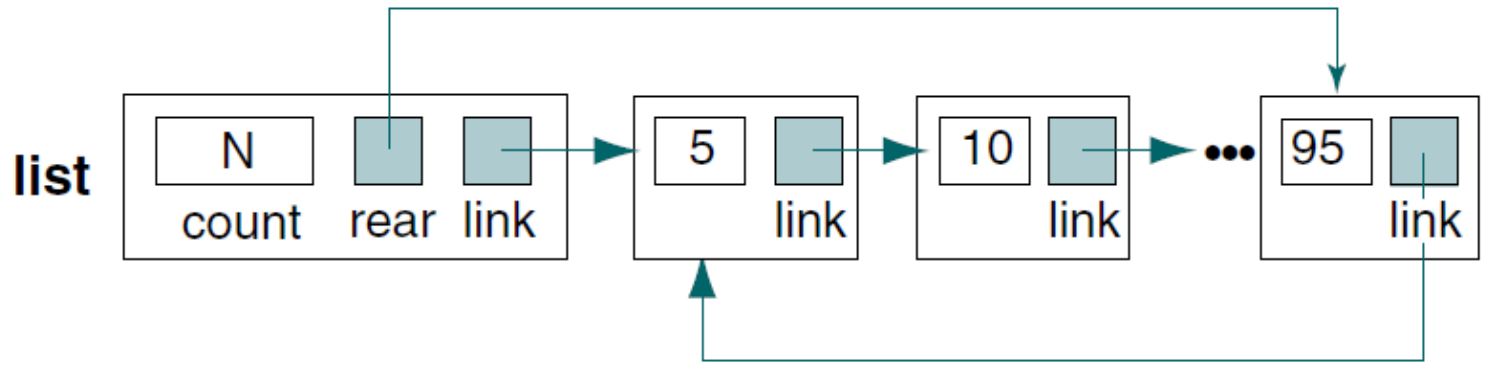
Complex Linked List Structure

- Üç tane kullanışlı bağlı liste çeşidi:
  1. **Circularly linked list** (Dairesel bağlı liste)
  2. **Doubly linked list** (Çift bağlı liste)
  3. **Multilinked list** (Çok bağlı liste)

Complex Linked List Structure
Circularly Linked List

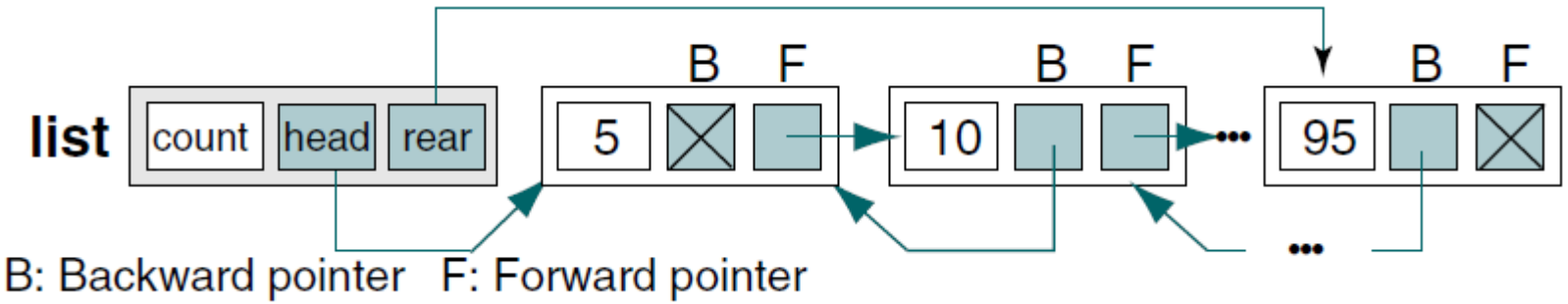
- Son düğümdeki bağlantı ilk düğüme işaret eder. Ayrıca başlık düğümüne (**header node**) de işaret edebilir.
- Ekleme ve silme, son düğümün ilk düğümü göstermesi dışında tek bağlı (**singly linkled**) listeyle aynıdır.
- Search problem: arama ne zaman/nasıl sonlandırılacak?
  - **Çözüm**: başlangıç düğümünün adresini kaydederiz ve bunun etrafında bir tur döndüğümüzde dururuz.

```
loop (target <>pLoc→data.key AND pLoc→link <> startAddress)
```

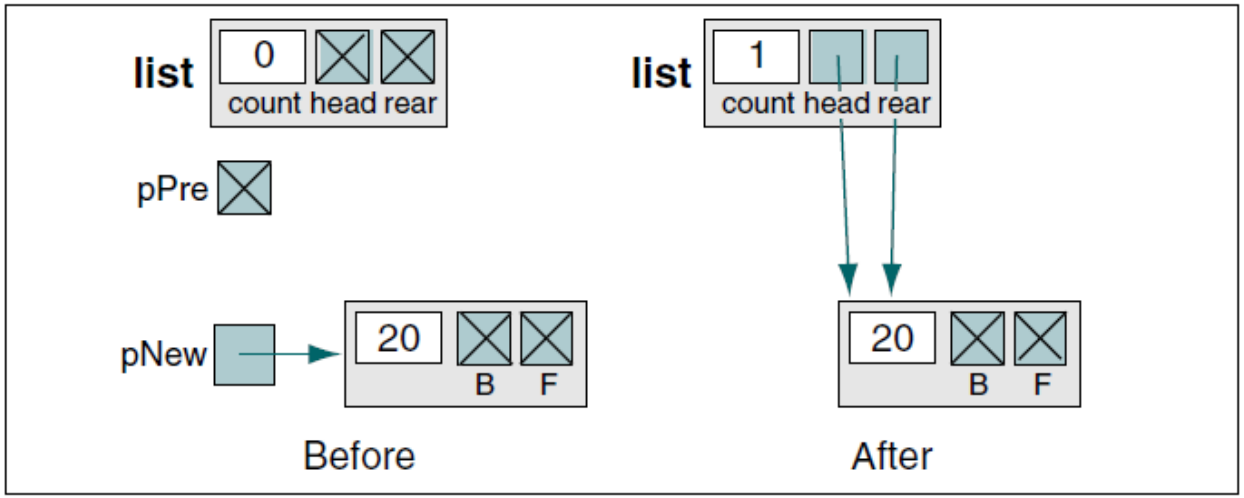


Complex Linked List Structure
Doubly Linked List (Çift bağlı liste)

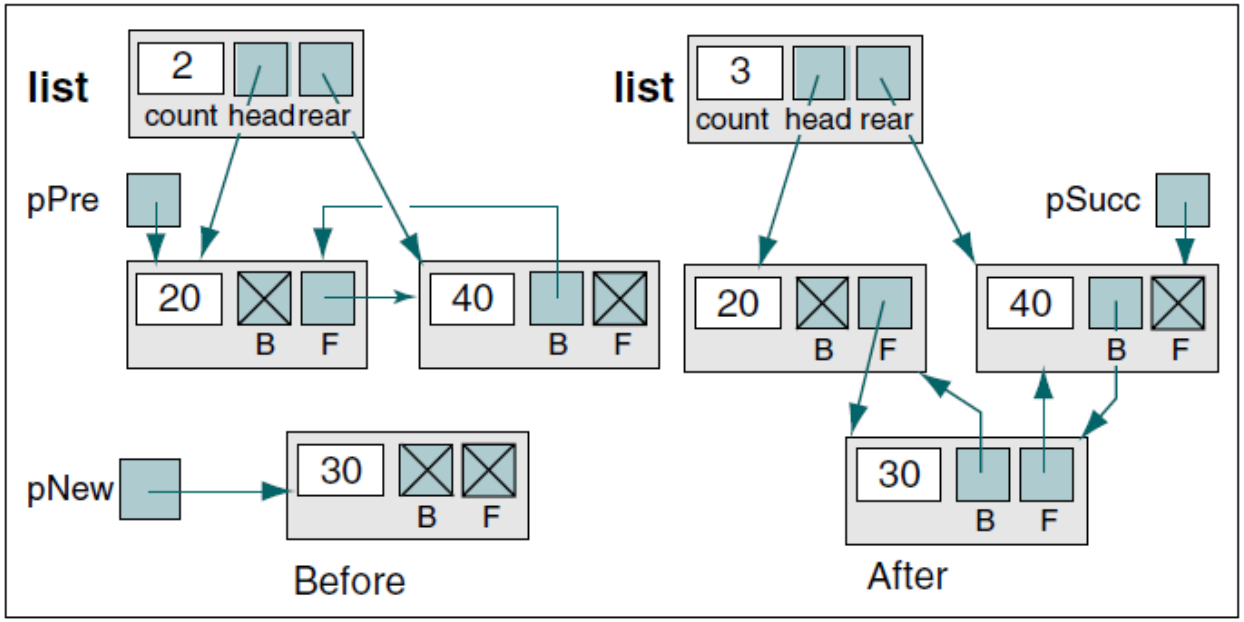
- Her düğümün hem önceki (selef) hem de sonraki (halef) düğüm için bir işaretçiye sahip olduğu bağlı bir liste yapısıdır.
  - Selefini gösteren bir geriye işaretçi (backward pointer to its predecessor)
  - Halefini gösteren bir ileri işaretçi (forward pointer to its successor)
- Çift bağlı listenin bir varyasyonu da "Çift bağlı dairesel liste «**doubly linked circularly linked list**» dir.



Doubly Linked List **-Insertion**



(a) Insert into null list or before first node



(b) Insert between two nodes

Doubly Linked List **-Insertion**

algorithm **insertDbl** (val pList <node pointer>, val dataIn <dataType>)

This algorithm inserts data into a doubly linked list.

PRE pList is a pointer to a valid doubly linked list.

dataIn contains the data to be inserted.

POST The data have been inserted in sequence

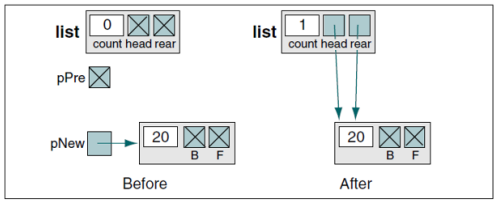
RETURN <integer> 0: failed– dynamic memory overflow

1: successful

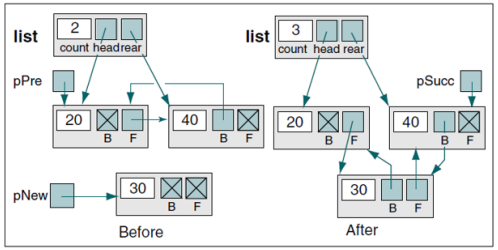
2: failed- duplicate key presented

Doubly Linked List -**Insertion**

```
1 if (full list)
    1 return 0
Locate insertion position in list (Eklenecek yeri belirleme)
2 found = searchList(pList, pPre, pSucc, dataIn.key)
3 if (found==false)
    1 allocate (pNew)
    2 pNew→userData = dataIn
    3 if (pPre==null)
        Inserting before first node or into empty list (İlk düğümden önce veya boş listeye ekleme)
        1 pNew→back = null
        2 pNew→fore = pList →head
        3 pList →head= pNew
    4 else
        Inserting into middle or end of list (listenin ortasına veya sonuna ekleme)
        1 pNew→fore=pPre→fore
        2 pNew→back = pPre
    Test for insert into null list or at end of list (boş listeye veya listenin sonuna ekleme testi)
    5 if (pPre→fore==null) –Inserting at end of list, set rear pointer
        1 pList→metadata.rear = pNew
    6 else – Inserting in middle of list, point successor to new
        1 pSucc→back = pNew
    7 pPre→fore = pNew
    8 pList→metadata.count = pList→metadata.count +1
    9 return 1
Duplicate data. Key already exists.
4 return 2
end insertDbl
```

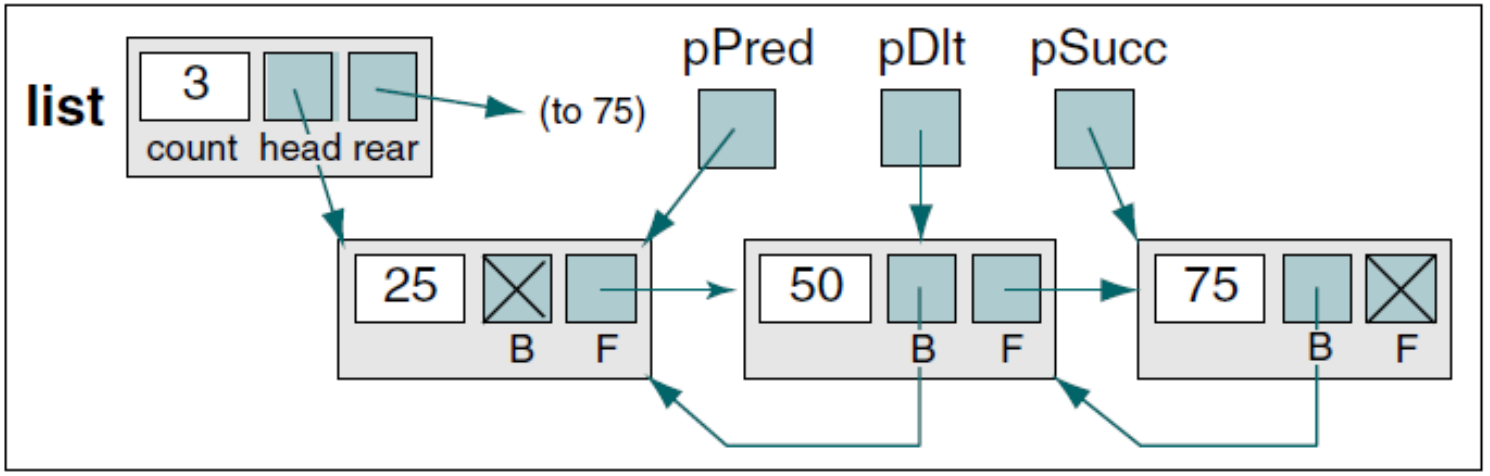


(a) Insert into null list or before first node

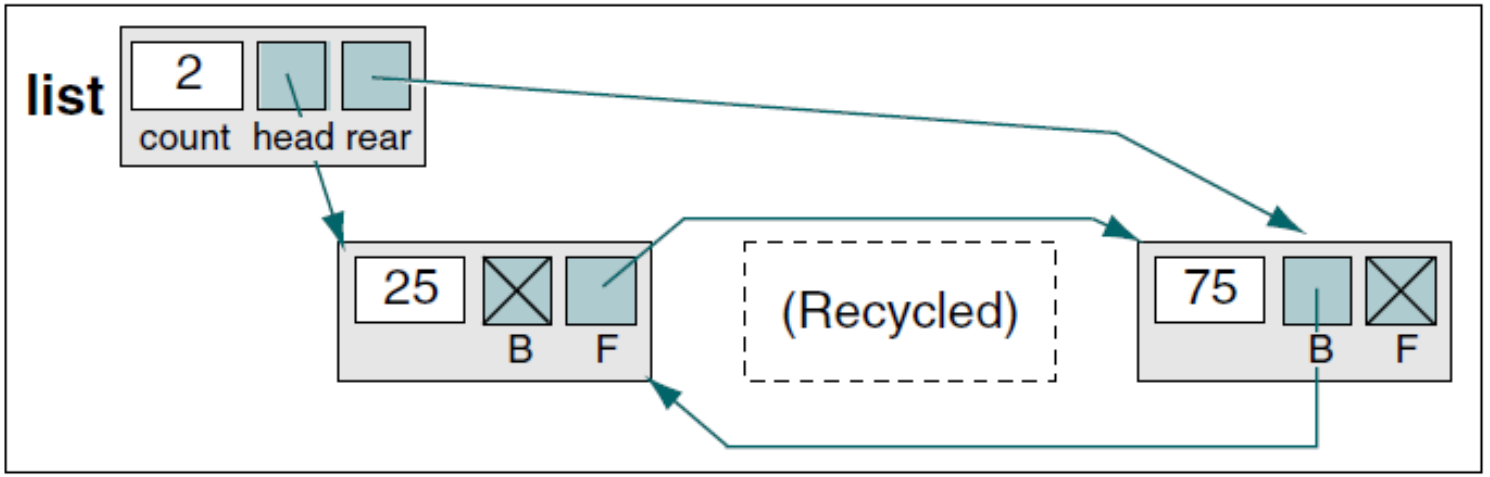


(b) Insert between two nodes

Doubly Linked List **-Deletion**



(a) Before delete



(b) After deleting 50

Doubly Linked List **-Deletion**

algorithm **deleteDbl** (val pList <node pointer>, val pDlt <node pointer>)

This algorithm deletes a node from a doubly linked list.

PRE pList is a pointer to a valid doubly linked list.

pDlt is a pointer to the node to be deleted.

POST node deleted

Doubly Linked List -**Deletion**

1 if (pDlt==null)

  1 abort

2 pList→metadata.count =pList→metadata.count – 1

3 if (pDlt→back<>null) Silinen eleman ilk düğüm değilse

  Point predecessor to successor

  1 pPred = pDlt→back

  2 pPred→fore = pDlt→fore

4 else

  Update head pointer

  1 pList→head=pDlt→fore

5 if (pDlt→fore<>null) Silinen eleman son düğüm değilse

  Point successor to predecessor

  1 pSucc = pDlt→fore

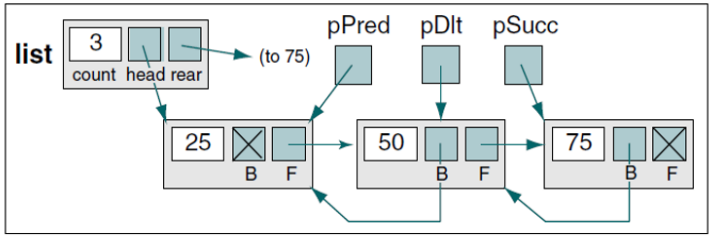
  2 pSucc→back = pDlt→back

6 else

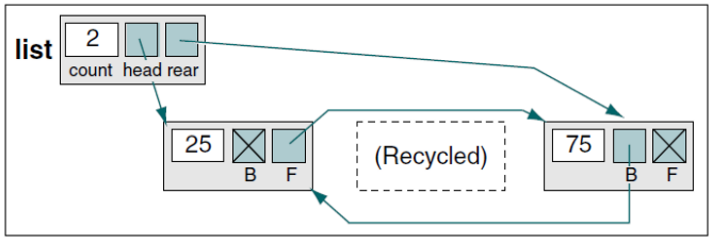
  1 pList→rear=pDlt→back

7 recycled pDlt

end **deleteDbl**



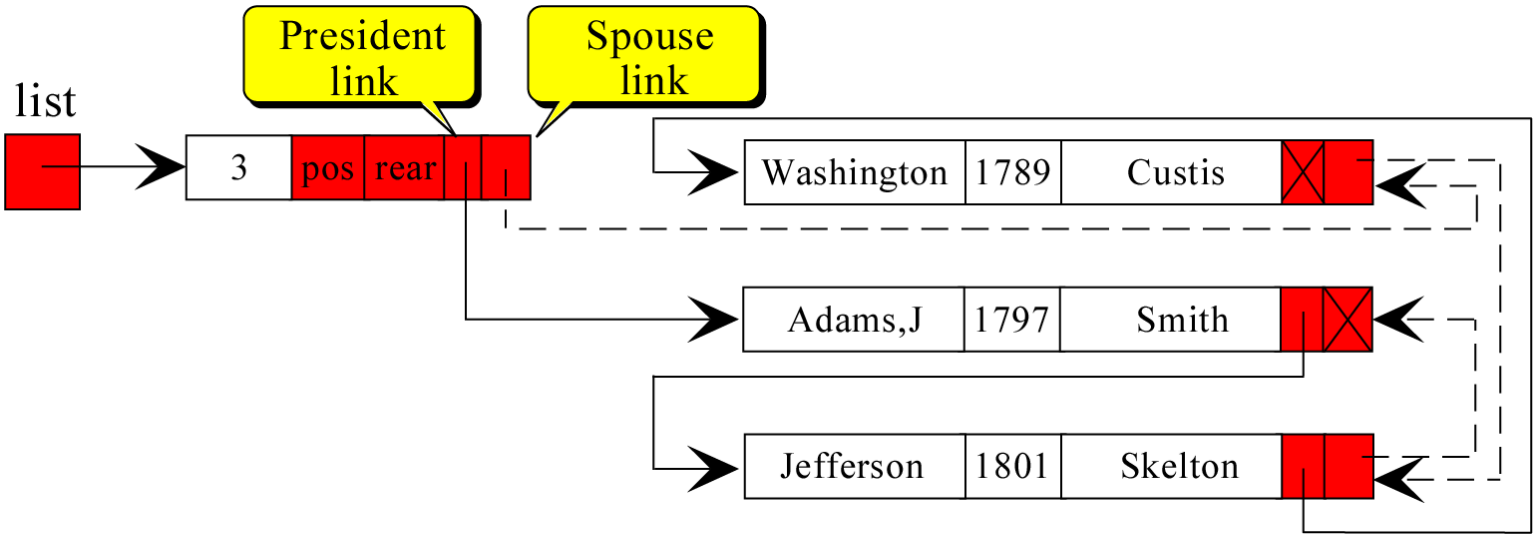
(a) Before delete



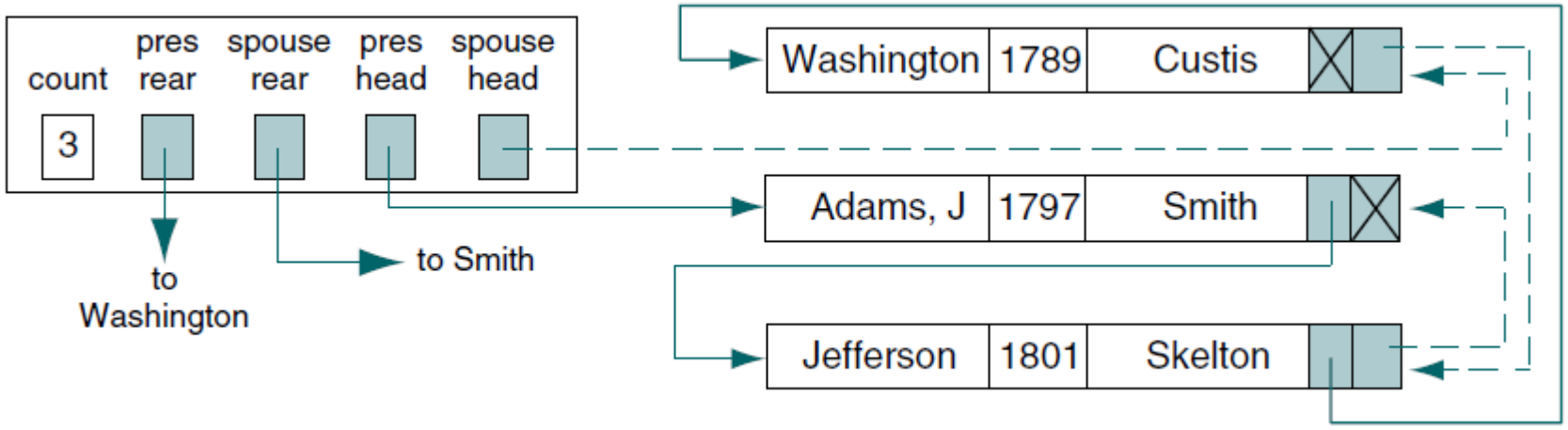
(b) After deleting 50

Complex Linked List Structure
Multilinked List

- İki veya daha fazla mantıksal anahtar dizilimi olan bir listedir.
- Veriler birden çok sırayla işlenebilir.
- Veriler çoğaltılmaz (not replicated)



Complex Linked List Structure
Multilinked List



President	Year	First lady
Washington, George	1789	Custis, Martha Dandridge
Adams, John	1797	Smith, Abigail
Jefferson, Thomas	1801	Skelton, Martha Wayles
Madison, James	1809	Todd, Dorothy Payne
Monroe, James	1817	Kortright, Elizabeth
Adams, John Quincy	1825	Johnson, Louisa Catherine
Jackson, Andrew	1829	Robards, Rachel Donelson
Van Buren, Martin	1837	Hoes, Hannah
Harrison, William H.	1841	Symmes, Anna
Tyler, John	1841	Christian, Letitia

TABLE 5-2 First 10 Presidents of the United States

Tablo 5-2'deki veriler, Amerika başkanlarının ilk göreve geldiği tarihe (yıl) göre kronolojik olarak listelenmiştir.

Bu datalar için iki ilave sıralama söz konusu olabilir:

- Başkanın ismine göre
- Eşinin ismine göre

Ödev 3

Öğrenci bilgilerini işlemek için çift bağlantılı (**doubly linked**) bir liste yapısı oluşturun. Öğrenci numarası anahtar bilgi olmalıdır.

Aşağıdaki gibi bir menü kontrolü altında çalışacak şekilde C kodunuzu oluşturun:

- Create link list
- Destroy linked list
- Add node
- Delete node
- Search node
- Display list (traverse list)

Öğrenci No
Ad
Soyad
Bölüm
Sınıf
...