BMT207
Veri Yapıları

Günay TEMÜR
Düzce Üniversitesi
Bilgisayar Mühendisliği

BÖLÜM - 4

İçerik;

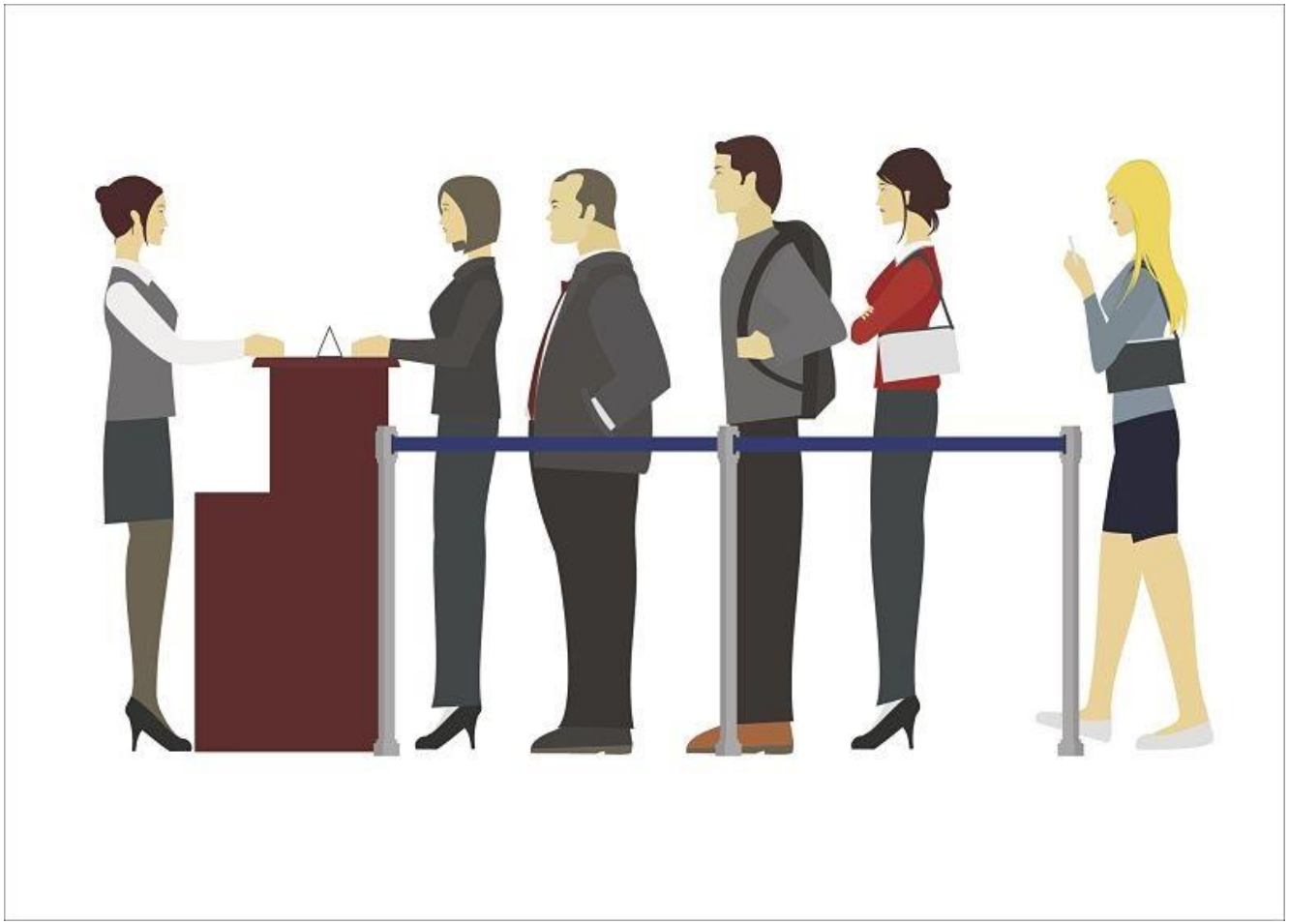
- Kuyruk Veri Yapısı Modeli
- Basit Kuyruk (Simple Queue)
- Döngüsel Kuyruk

Kuyruk Giriş

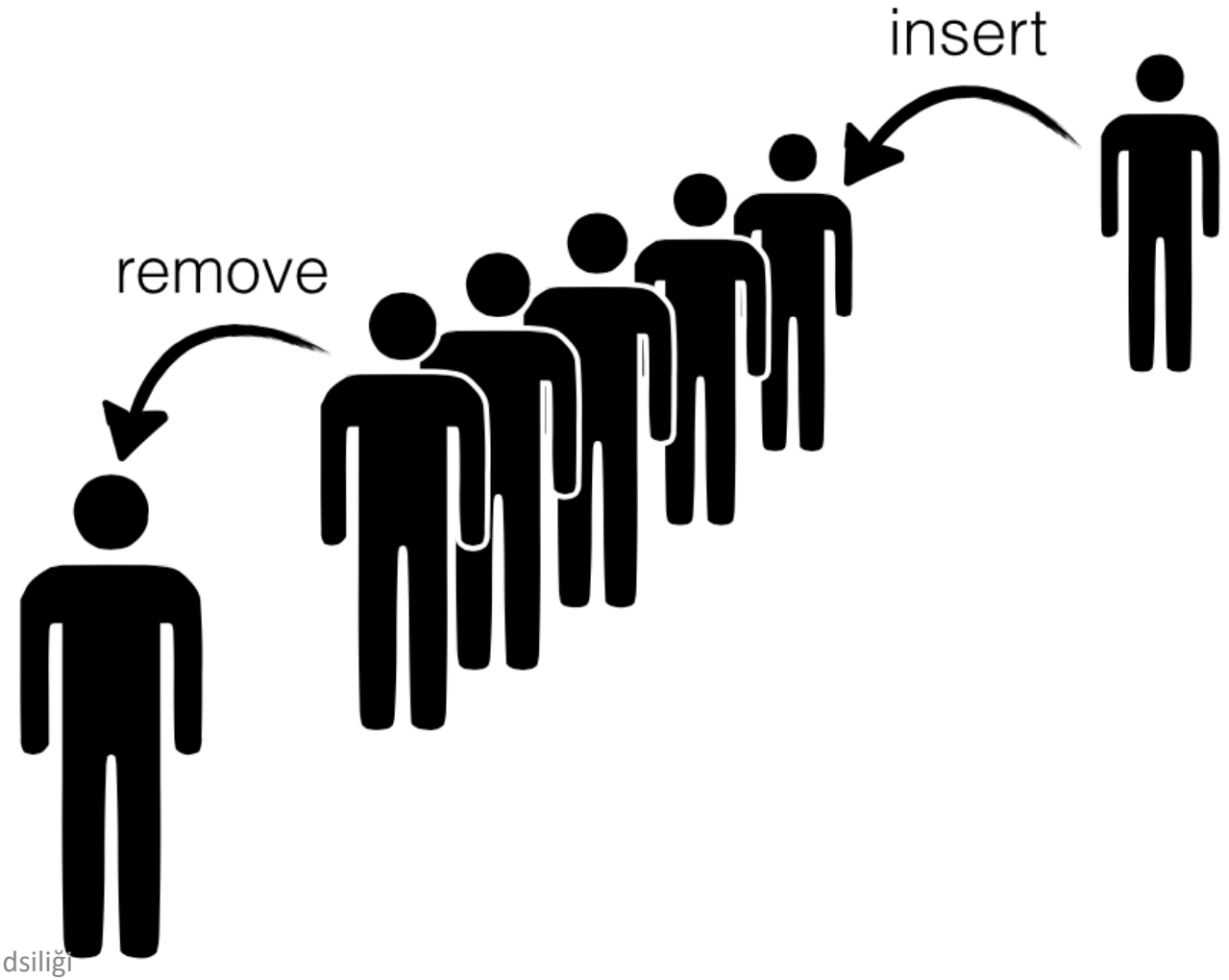
•Kuyruk, eleman eklemelerin sondan (**rear**) ve eleman çıkarmaların bastan (**front**) yapıldığı,
•(First In First Out-İlk Gelen İlk Çıkar – FIFO) olarak modellenen, doğrusal bir veri yapısı modelidir.
•Bir elemanın kuyruğa girmesi **insert** (*enqueue)* işlemi iken listeden silinmesi **remove** (*dequeue*) işlemidir.
•Insert'ler *kuyruğun arkasından yapılırken*, remove'lar *kuyruğun önünden yapılırlar*.
•**Boş bir kuyruktan** eleman silmeye çalışmak ***underflow*** hatası üretirken,
•**dolu bir kuyruğa** eleman eklemeye çalışmak ***overflow*** hatası üretir.

Kuyruk

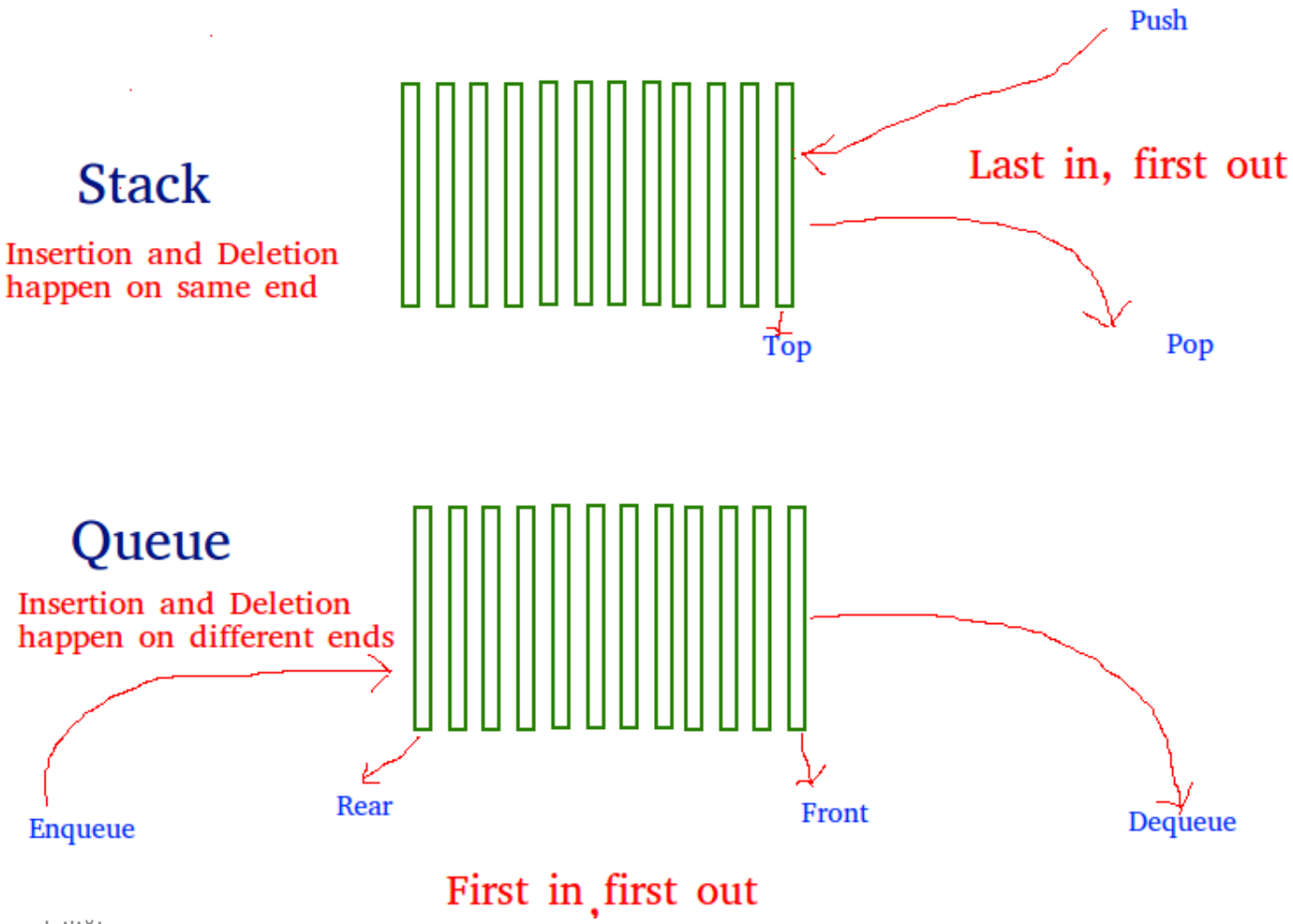
Kuyruk



Kuyruk



Kuyruk & Stack



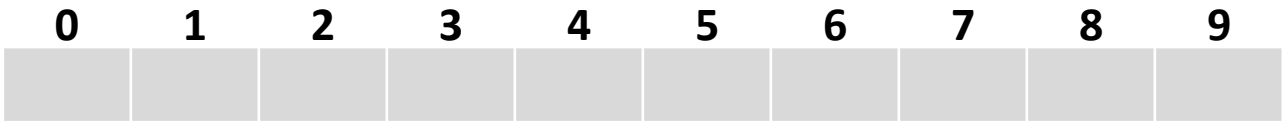
Queue Çalışma Mantığı

Queue dizi implementasyonu için kurallar

- Implementasyonda **front** ve **rear** olmak üzere 2 tane değişken tanımlanır.
  - **front:** kuyruğun önündeki elemanı temsil eder.
    - front = -1 ise kuyruk boştur.
    - Kuyruktan her eleman çıkartıldığında (REMOVE) front bir artar.
  - **rear:** kuyruğun sonundaki elemanı temsil eder.
    - Kuyruğa her eleman eklendiğinde (INSERT) rear bir artar.

Queue Çalışma Mantığı (devam…)

front=-1



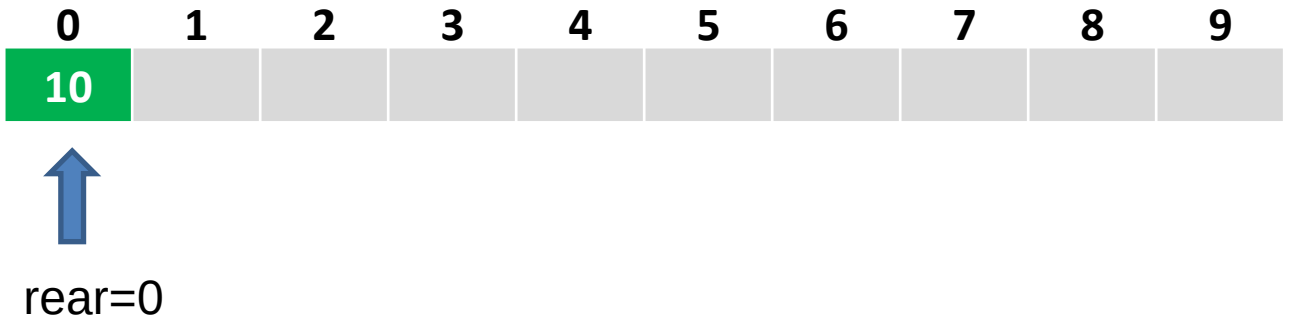
rear=-1

front ve rear

- Kuyruğa bir eleman **eklenince** ne olur?
- Kuyruktan bir eleman **çıkartılınca** (işi bitince ne olur?)

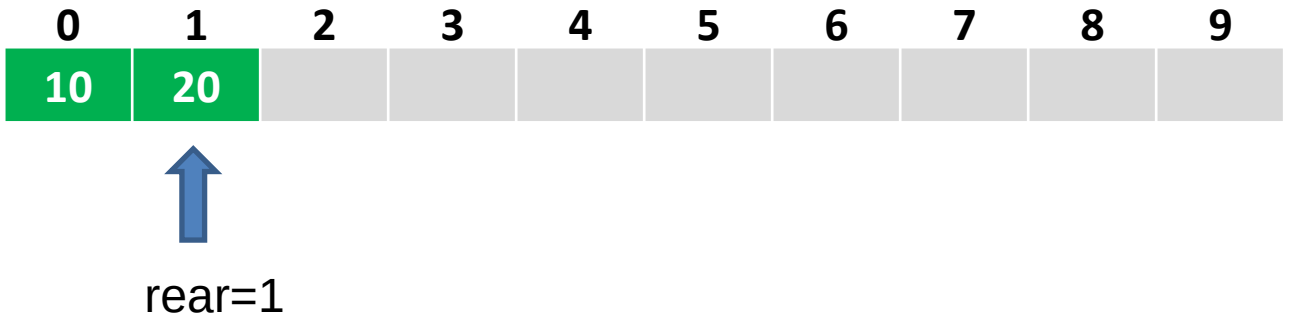
Queue Çalışma Mantığı (devam…)

front=-1



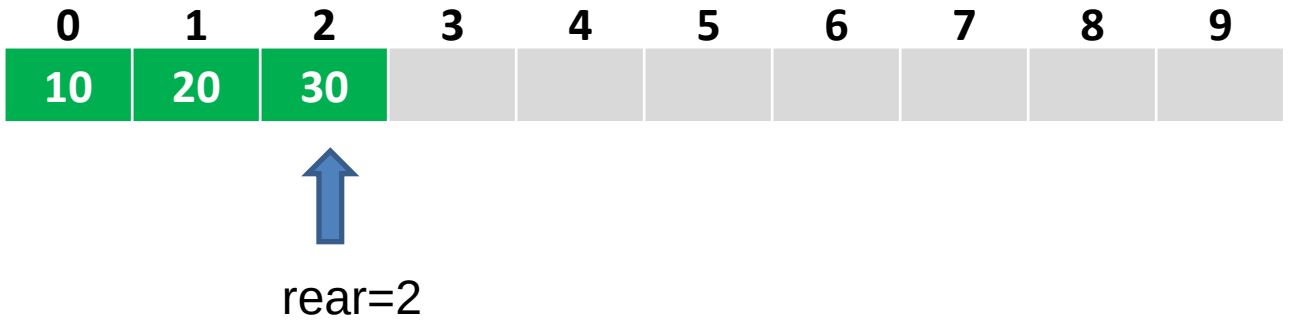
Queue Çalışma Mantığı (devam…)

front=-1



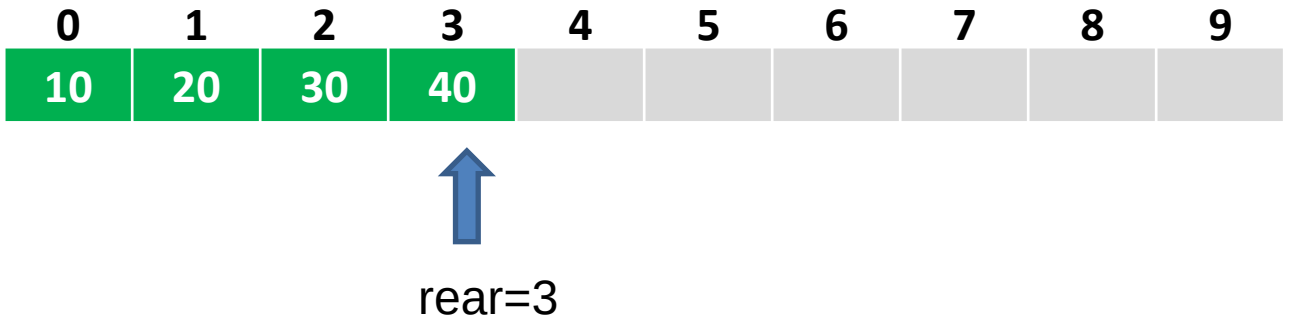
Queue Çalışma Mantığı (devam…)

front=-1



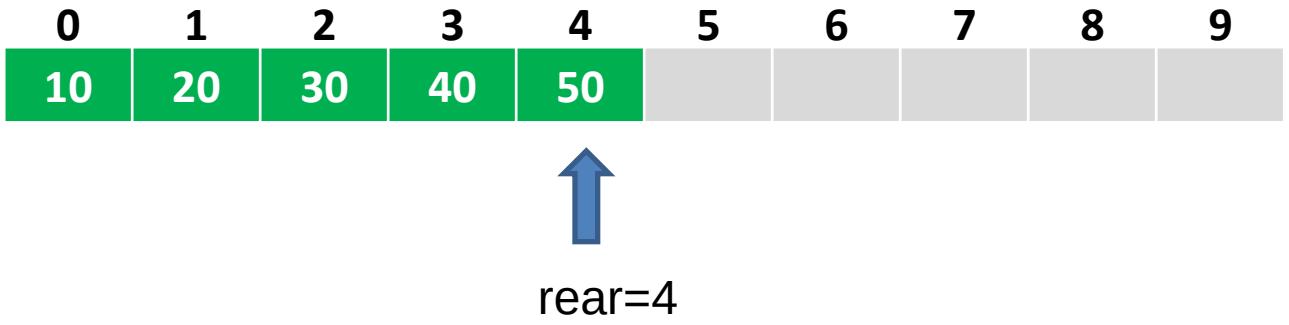
Queue Çalışma Mantığı (devam…)

front=-1

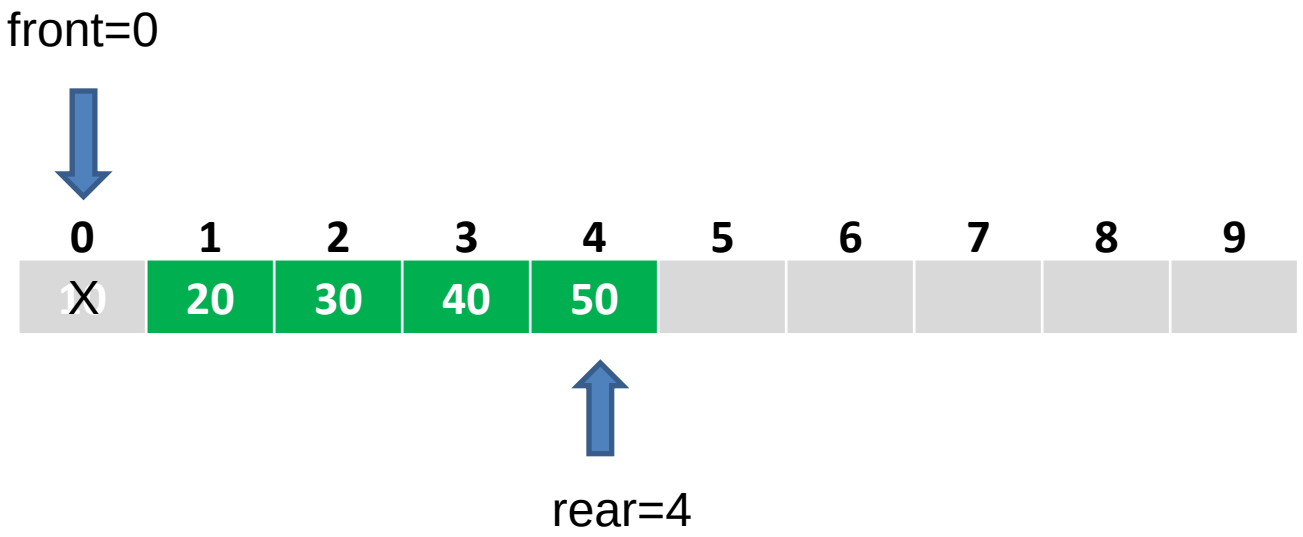


Queue Çalışma Mantığı (devam…)

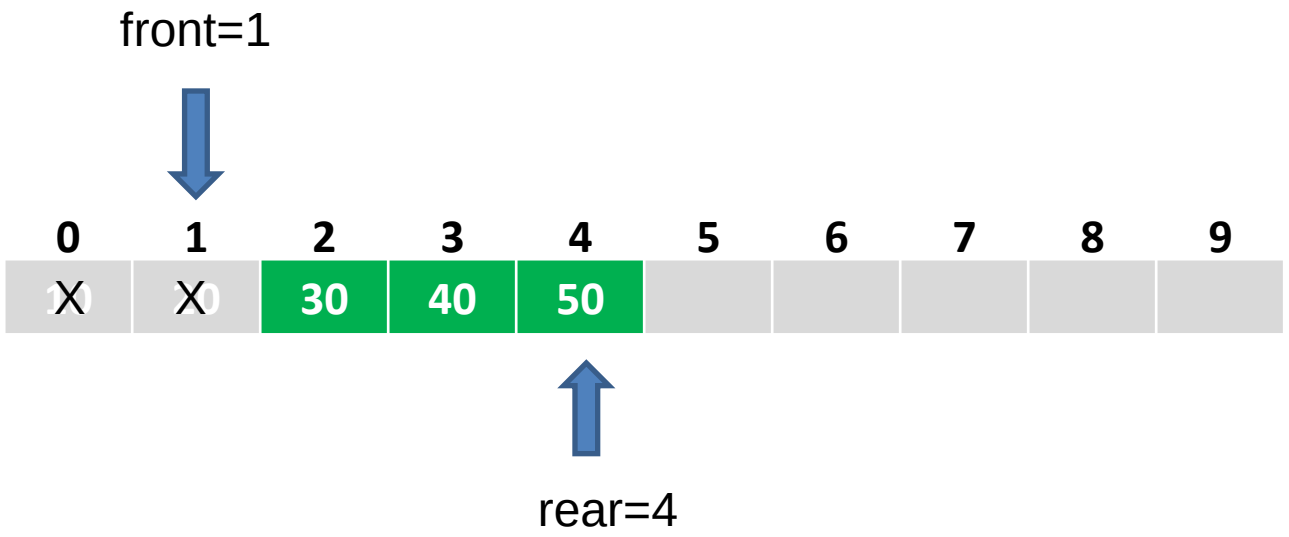
front=-1



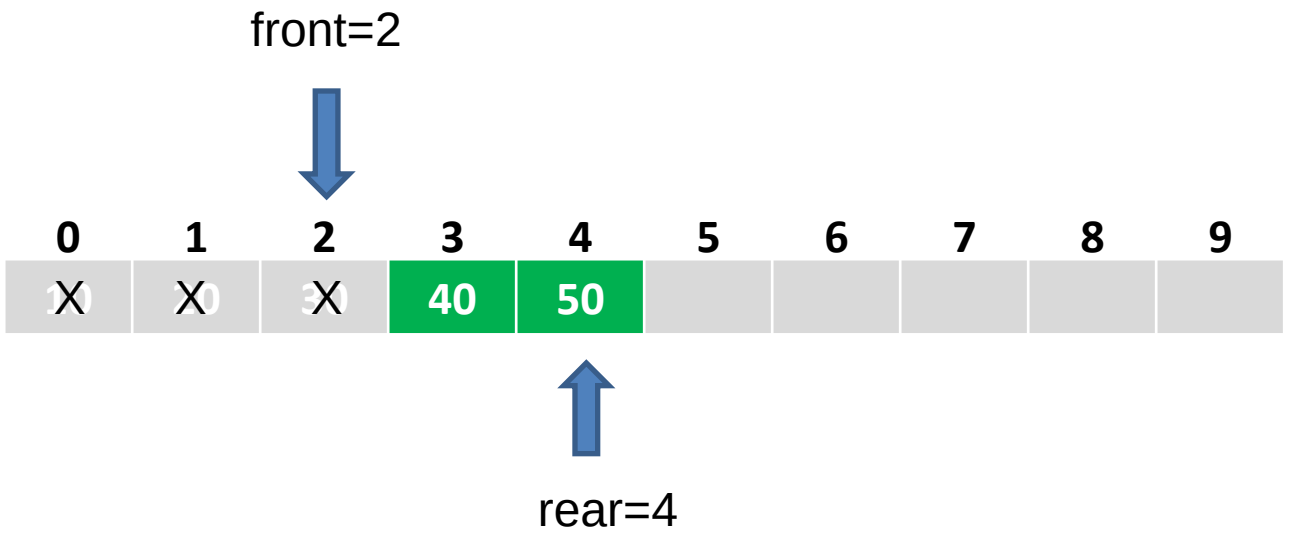
Queue Çalışma Mantığı (devam…)



Queue Çalışma Mantığı (devam…)



Queue Çalışma Mantığı (devam…)



Simple Queue

- Simple Queue (**basit kuyruk**) olarak adlandırılan bu kuyruk tipi
  - *hep ileri yönde hareket etmekte* ve
  - *verimsiz alan kullanımı*na neden olmaktadır.

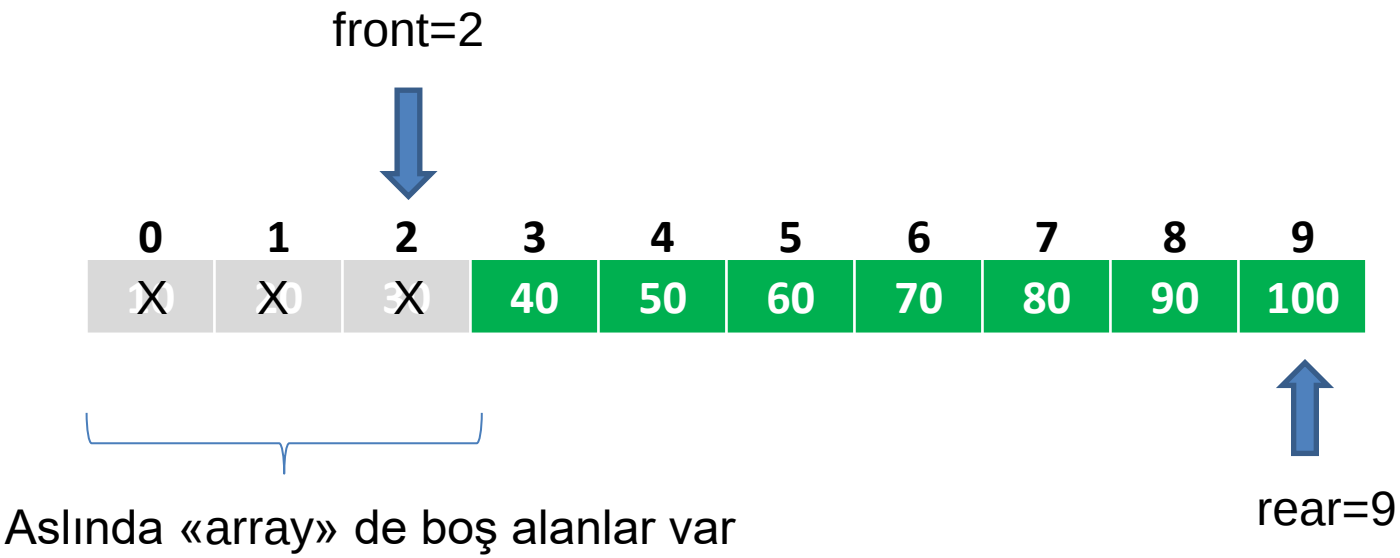
Simple Queue İyileştirmeleri

Basit kuyruk gerçekleştiriminde *çeşitli iyileştirmeler* yapılabilir:

- **İyileştirme 1:** Silme sonucunda kuyrukta hiç eleman kalmazsa, kuyruk sıfırdan oluşturulmuş gibi ***ilk durumuna*** getirilebilir.
- **İyileştirme 2:** Kaydırma (**shift**) işlemi yapılarak öndeki boş yerler kullanıma sokulmak üzere arkaya taşınabilir, fakat kaydırmalar aşırı zaman alır ve maliyetlidir.
- **İyileştirme 3:** Diğer iyileştirme kuyruğun boşta kalan öndeki alanlarını kullanmaya yönelik bir geliştirme yapılabilir (**Circular -** Döngüsel Kuyruk).

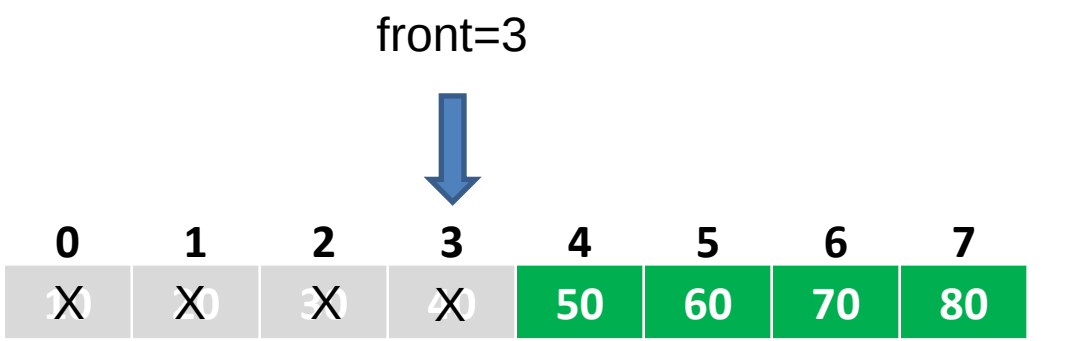
Circular Queue

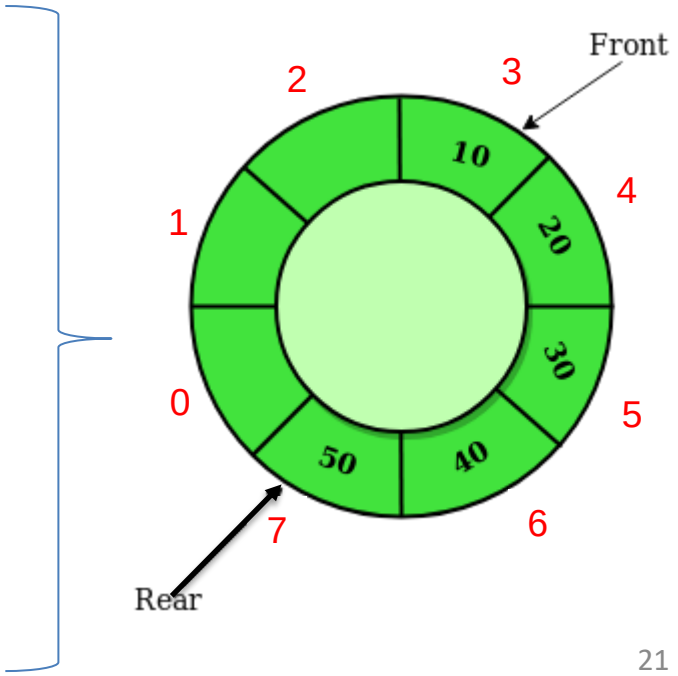
- Basit kuyrukta karşılaşılan ve kuyruğun başında kalan *kullanılamayan alan problemini çözmek için* döngüsel kuyruk veri yapısı modeli geliştirilmiştir.



Circular Queue

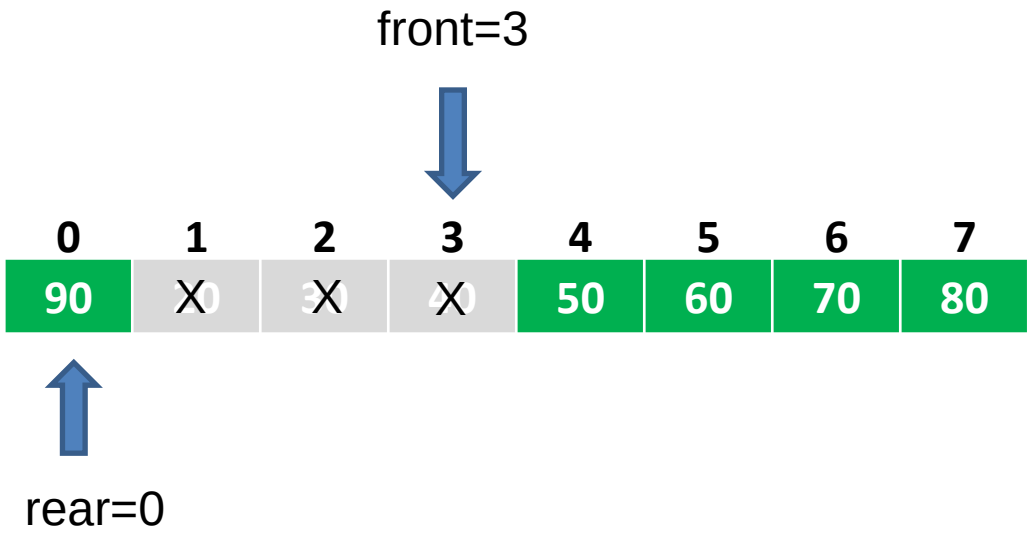
- Bu sebeple; Önde ***boş olan indisler***, arkadaymış gibi otomatik olarak kullanıma sokulur.





Circular Queue

- Öndeki ***boş olan indislere***, ekleme işlemi «array» dolana kadar devam edilir.

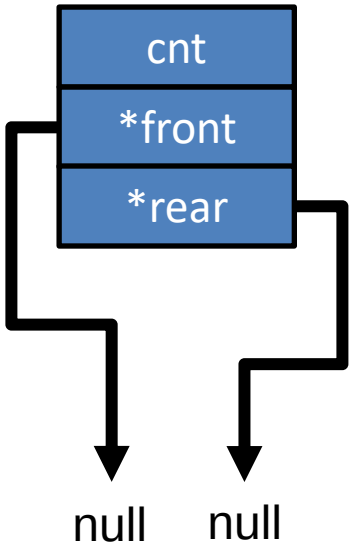


Circular Queue

- Ya! «array» de dolarsa ?

Queue Liste Implemantasyonu

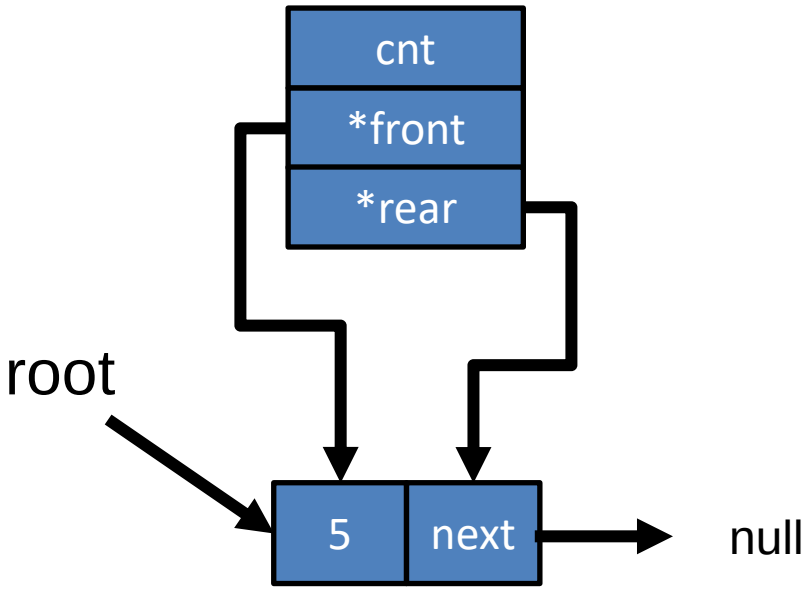
enqueue()




```
struct {
    int cnt;
    node *front;
    node *rear;
};
```

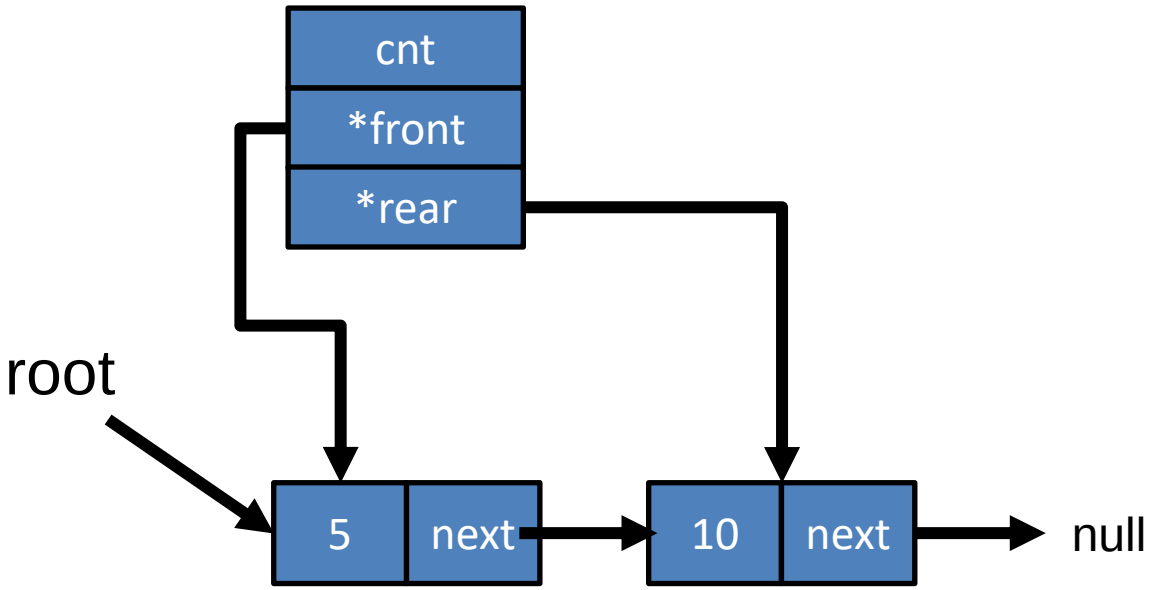
Queue Liste Implemantasyonu

enqueue()



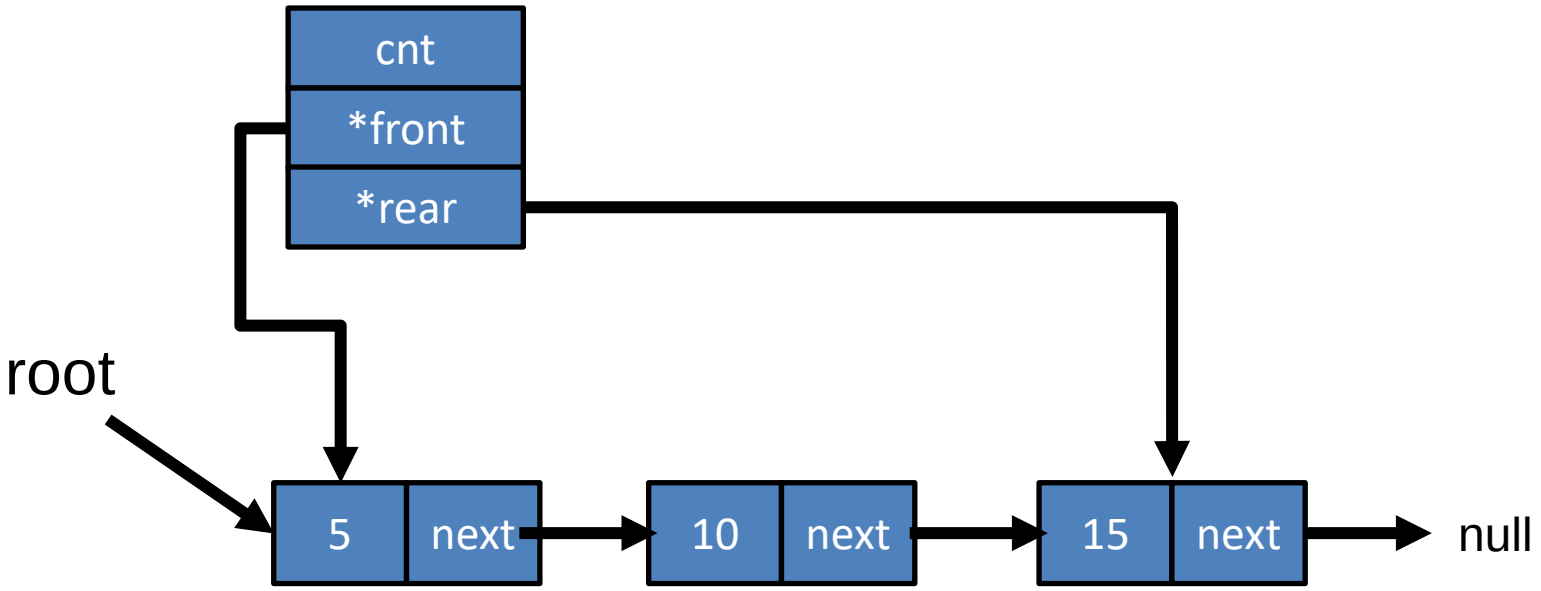
Queue Liste Implemantasyonu

enqueue()



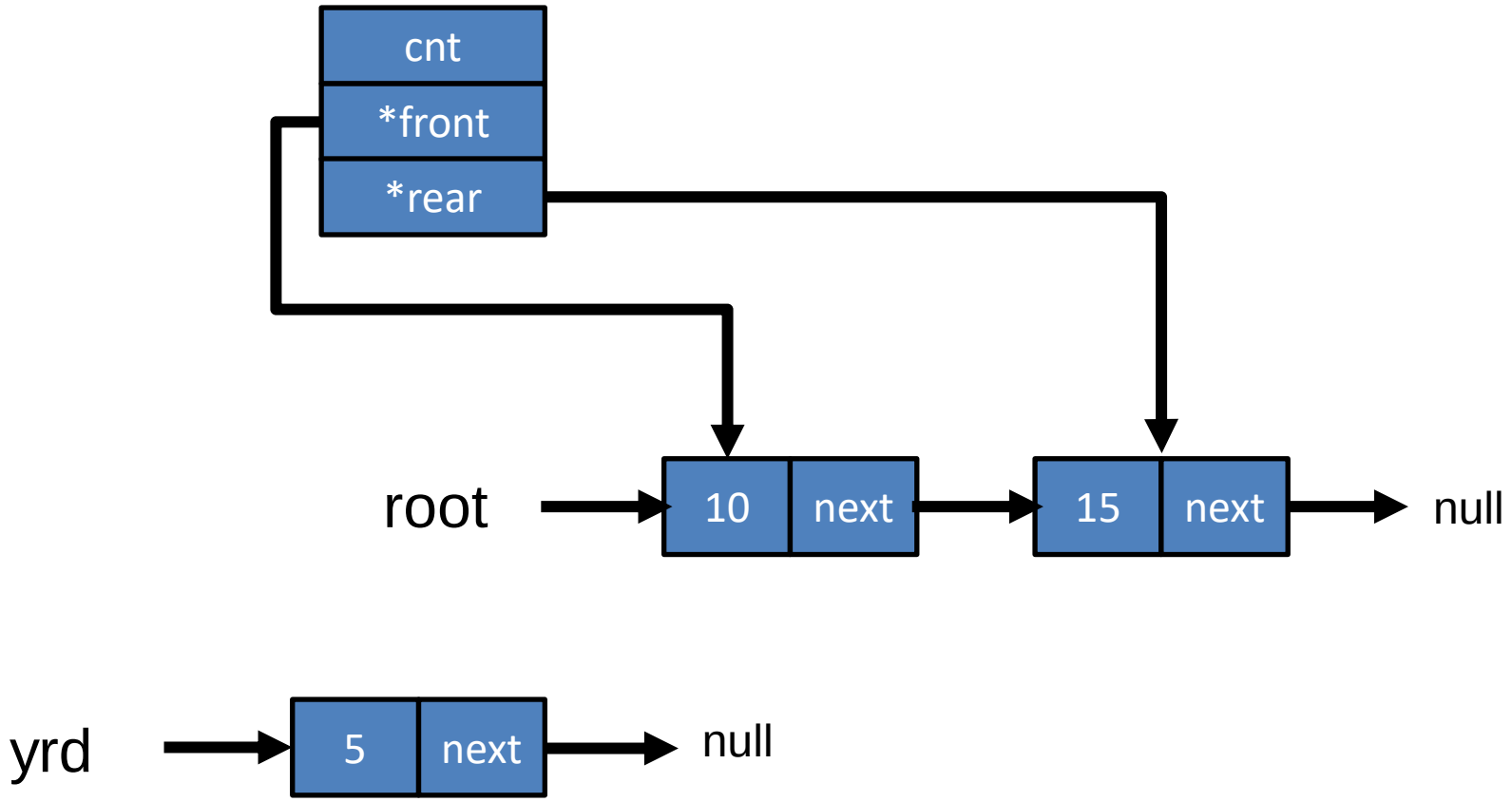
Queue Liste Implemantasyonu

enqueue()



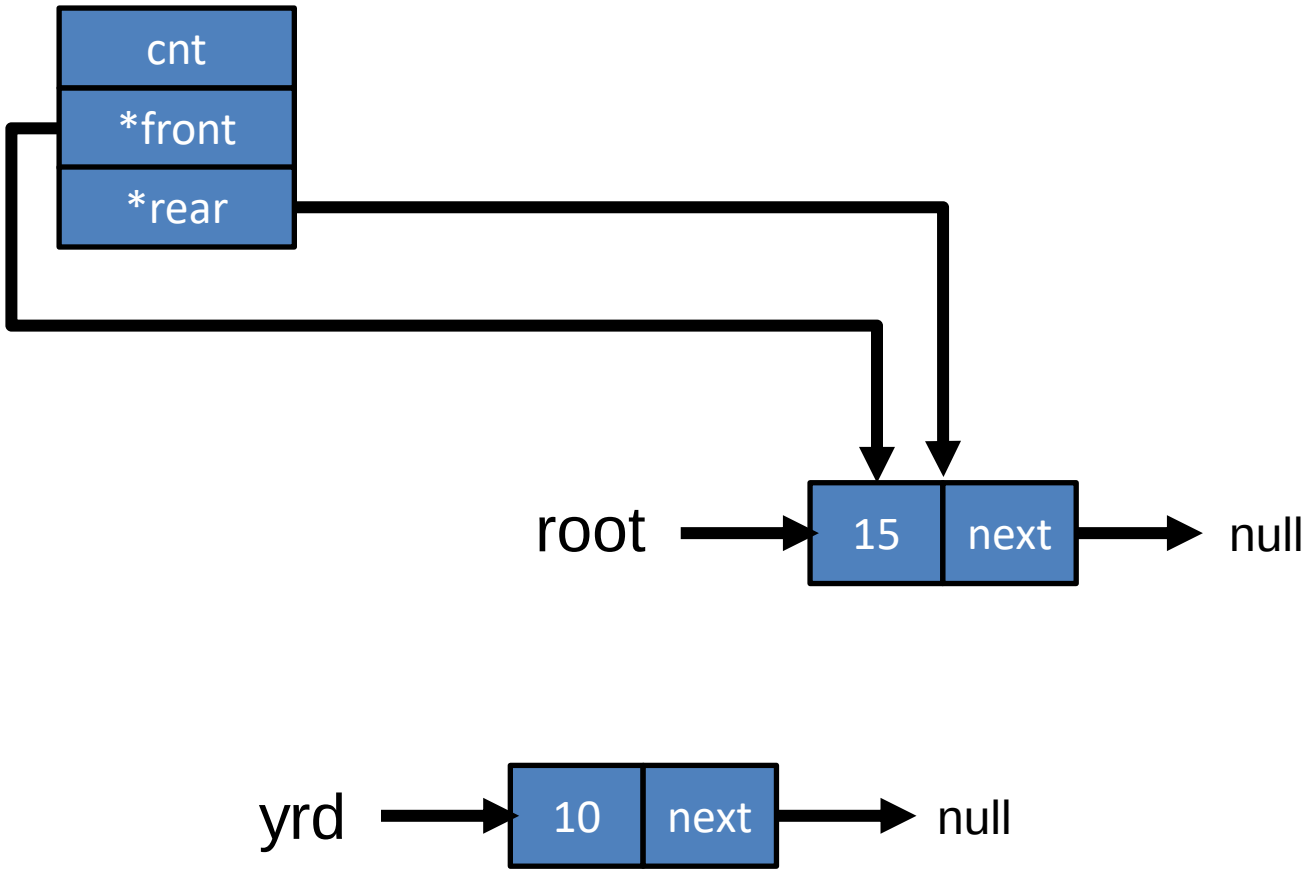
Queue Liste Implemantasyonu

dequeue()



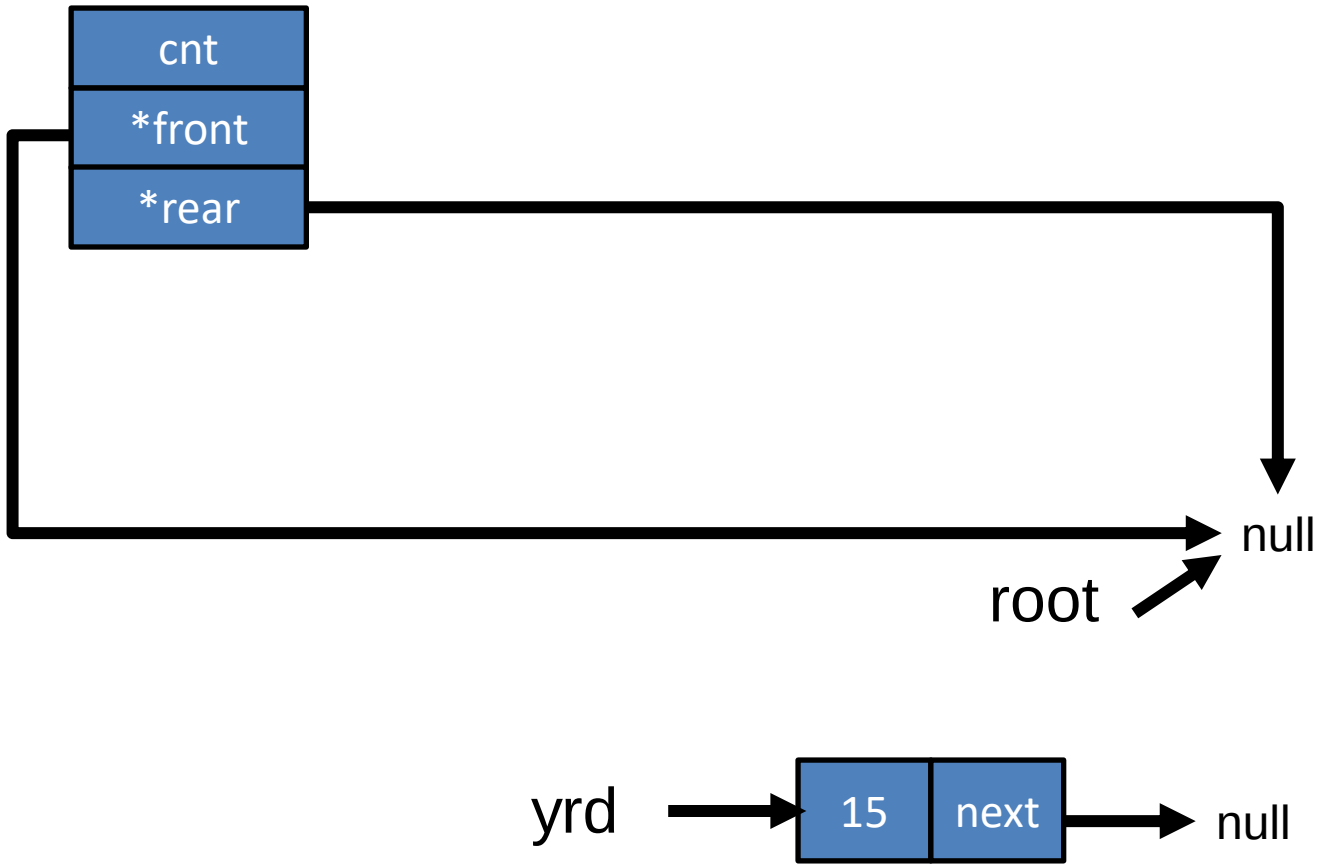
Queue Liste Implemantasyonu

dequeue()



Queue Liste Implemantasyonu

dequeue()



BİTTİ ☺

Yararlanılan Kaynaklar

- **Ders Kitabı:**
  - Veri Yapıları Rifat ÇÖLKESEN
  - Data Structures Using C, Reema Thareja
- **Yardımcı Okumalar:**
  - Celal Bayar Üniversitesi, Yrd. Doç. Dr. Deniz KILINÇ hocanın sunumları.