FTI

TEKNIK ELEKTRO

EIE5023

ALGORITMA DAN STRUKTUR DATA + PRAK

5

ENDANG SRI RAHAYU

Outlines:

QUEQUE :

- Konsep FIFO
- Operasi Dasar queque
- Priority Queue
- Penggunaan Queue

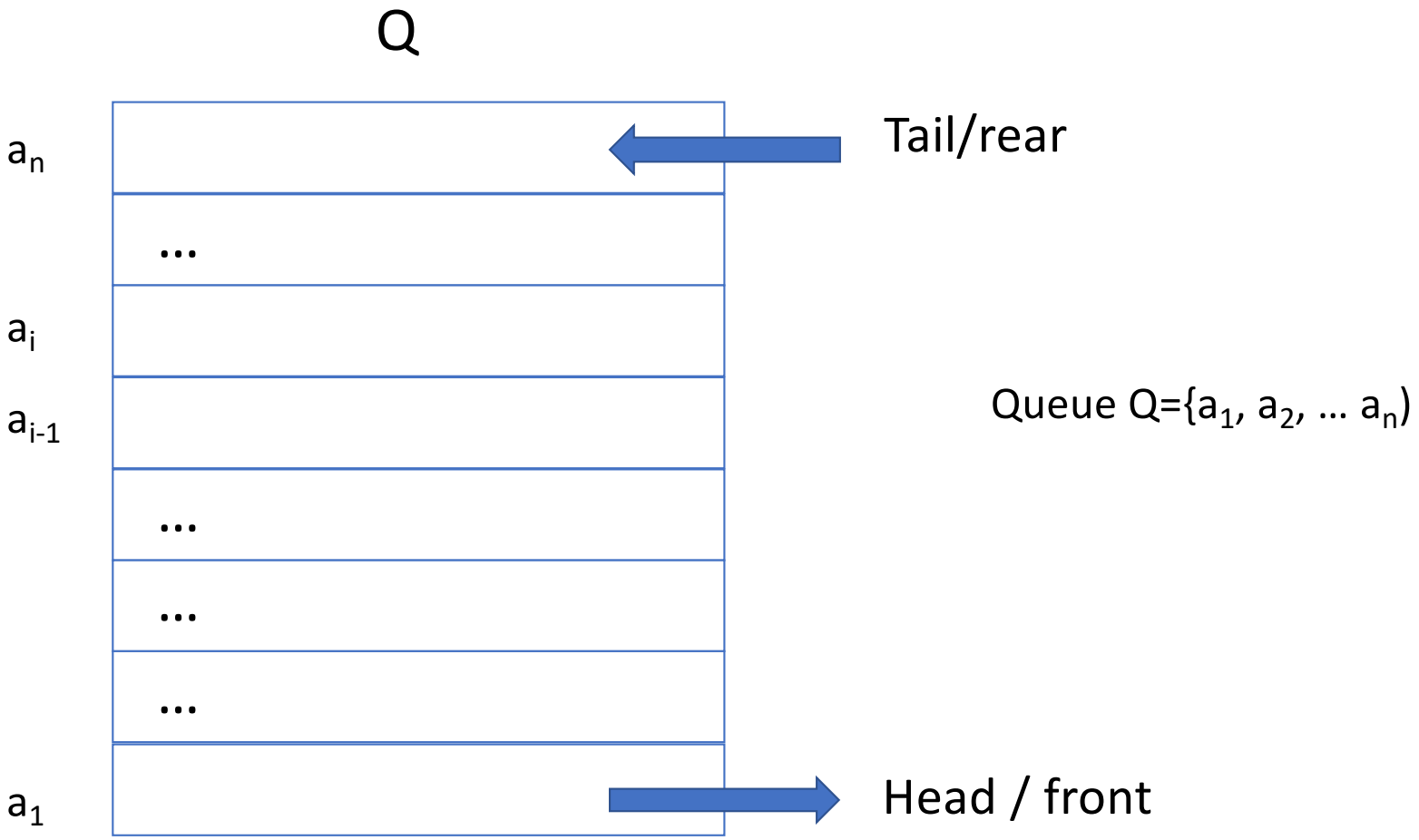
Endang Sri Rahayu

ALGORITMA DAN STRUKTUR DATA

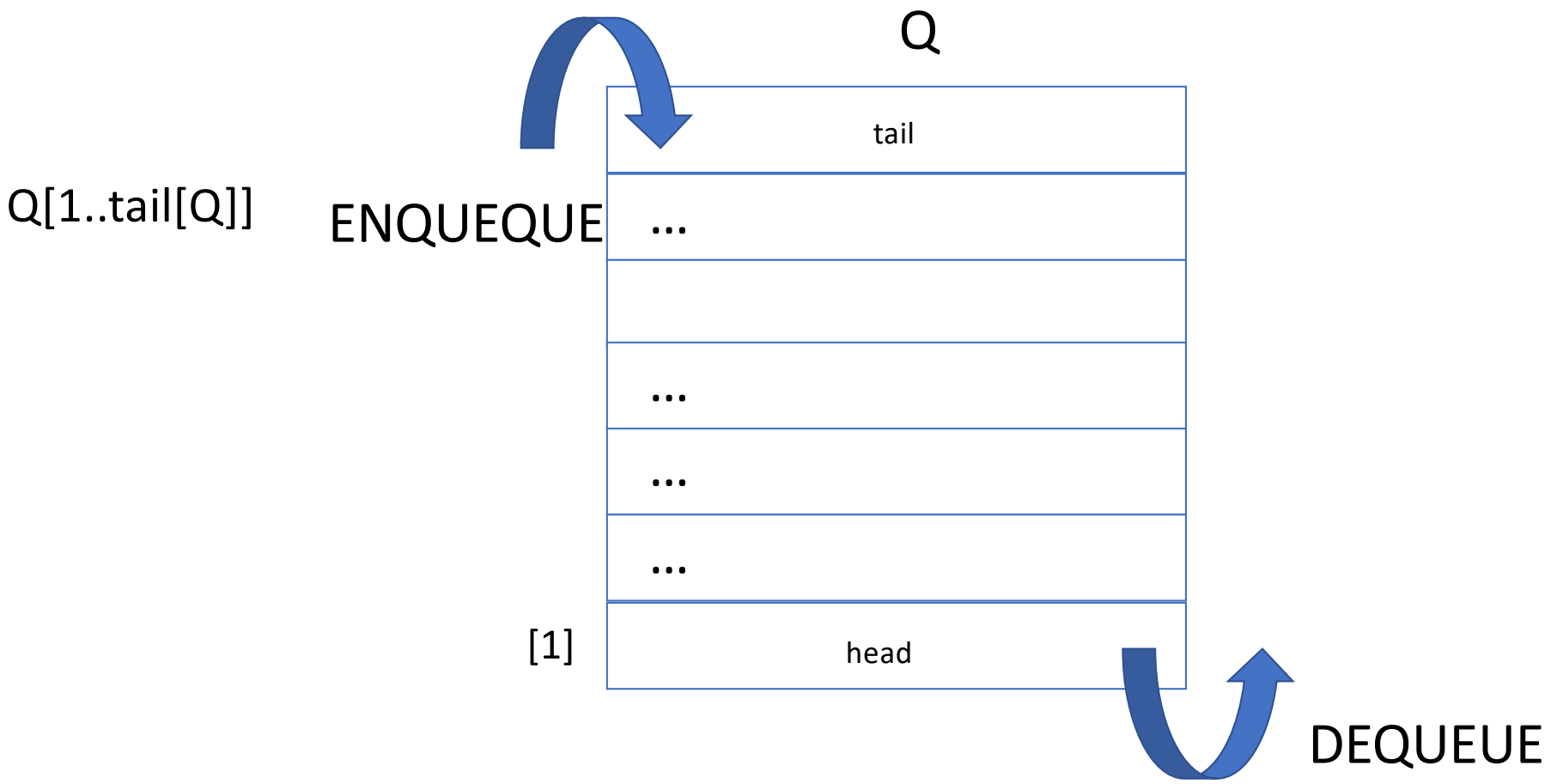
QUEUE

- ***Queue*** adalah struktur data linear dimana penambahan komponen dilakukan di satu ujung, sementara pengurangan dilakukan di ujung lain (yang satu lagi).
- Pada queue berlaku prinsip FIFO (*Last In First Out*), yang pertama masuk akan menjadi yang pertama dikeluarkan.

- ***Manfaat:***
  - Digunakan *operating* system untuk mengatur eksekusi dalam suatu sistem untuk mencapai perlakuan yang "adil" (seringkali *queue* disebut *waiting line*)
  - Untuk *mailbox* dalam komunikasi antar proses
  - Untuk *buffe*r dalam mekanisme *print spooler*, komunikasi data
  - Untuk simulasi dan modeling (misalnya simulasi sistem pengendali lalu lintas udara) dalam memprediksi *performance*.


Q
a_n
...
a_i
a_{i-1}
...
...
...
a_1
Tail/rear
Head / front
Queue Q={a_1, a_2, … a_n)

OPERASI DASAR QUEUE



Algoritma Enqueue dan Dequeue

ENQUEUE(Q,x)

1. Q(tail[Q]) ← x
2. **if** tail[Q] = length[Q]
3. **then** tail[Q] ← 1
4. **else** tail[Q] ← tail[Q] + 1
5. **return**

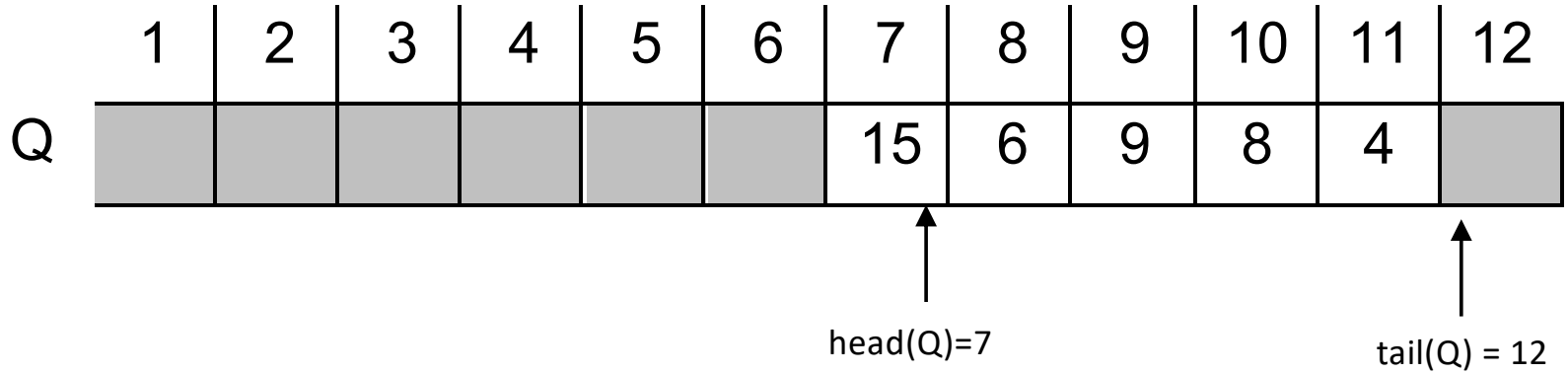
DEQUEUE

1. x ← Q[head[Q]]
2. **if** head[Q] = length[Q
3. **then** head[Q] ← 1
4. **else** head[Q] = head[Q] + 1
5. **return**

contoh operasi yang bisa dilakukan pada queue

- Operasi menciptakan Q dengan *queue* kosong, **InitQ(Q)**
- Operasi menambah elemen x ke **rear** *queue*, **AddQ(Q,x)** atau Procedure Push
- Operasi penghilangan elemen pada **front** dari *queue* Q , **DeleteQ(Q,x)** atau Procedure Pop
- Operasi mengirim elemen **front** dari *queue* Q, **frontQ(Q)**
- Operasi yang mengembalikan true if Q kosong else false, **isEmptyQ(Q)**
- Operasi mengirimkan jumlah elemen di *queue*, **howManyInQ(Q)**

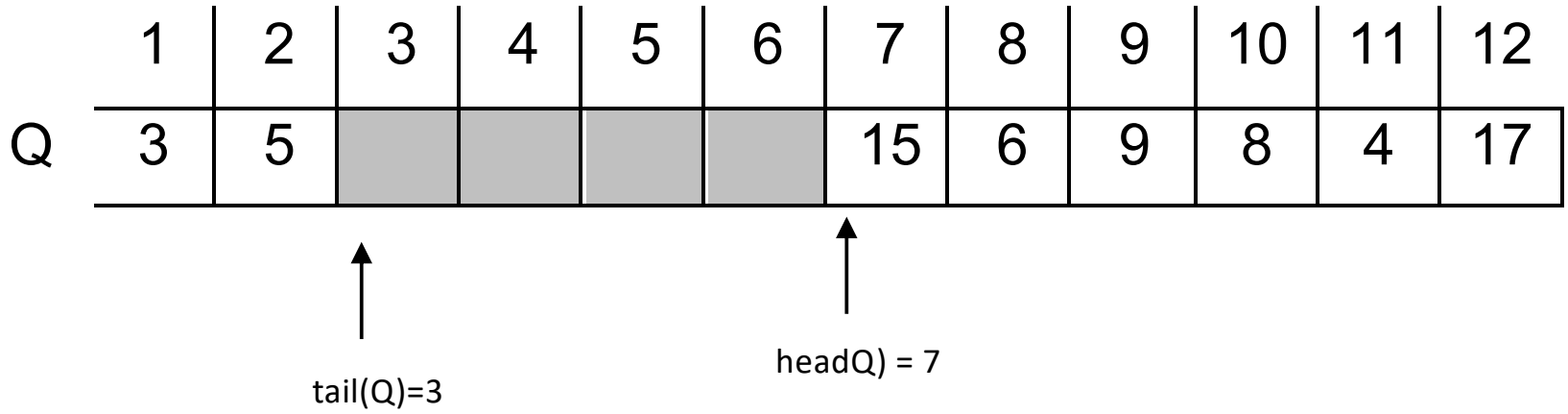
Implementasi queue dengan menggunakan array



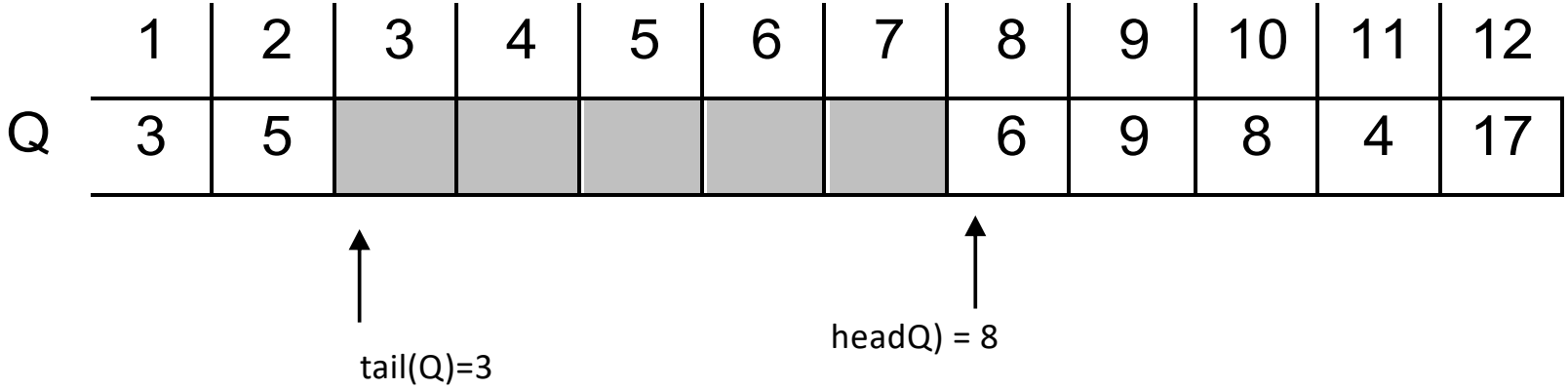
Dengan array Q[1..n] dapat menampung n-1 elemen. *Queue* memiliki atribut head[Q] yang berindeks dan menunjuk ke posisi head. Atribut tail[Q] berindeks dan menunjuk ke lokasi berikutnya dimana elemen baru disisipkan kedalam queue. Lokasi-lokasi elemen pada *queue* adalah head[Q], head[Q]+1, head[Q]+2, …, tail[Q] – 1.

Queue diimplementasikan menggunakan *array* Q[1..12]. Elemen-elemen *queue* hanya muncul pada posisi-posisi yang diberi warna terang.

Queue diatas memiliki 5 elemen pada lokasi Q[7..11]



Konfigurasi queue setelah dilakukan operasi Enqueue(Q,17), Enqueue(Q,3) dan Enqueue(Q,5) adalah sebagai berikut :

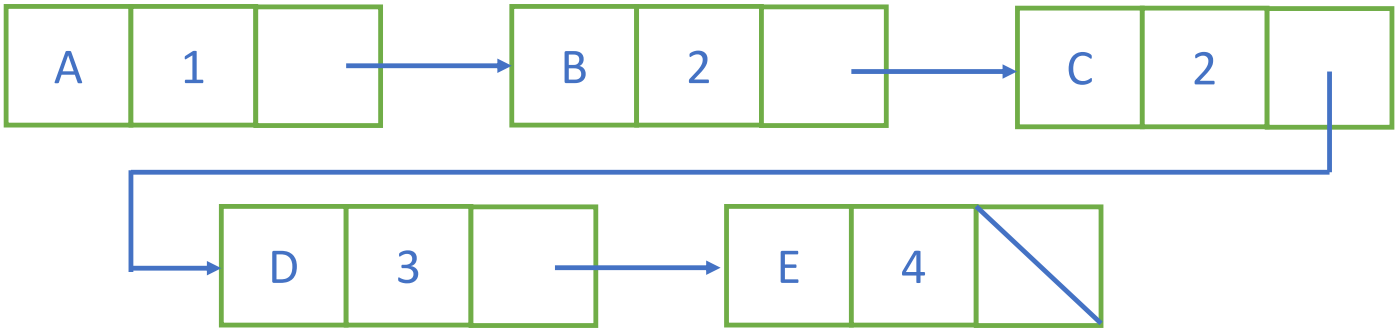


Konfigurasi *queue* setelah dilakukan operasi Dequeue(Q), yang mengambil nilai 15. head baru pada nilai 6.

PRIORITY QUEUE

- Merupakan kumpulan item data yang masing-masing memiliki tingkat prioritas yang bisa berbeda
- Apabila item data diambil dari kumpulan tersebut maka item data yang berprioritas paling tinggilah yang diperoleh.
- Setiap elemen yang akan masuk dalam queue sudah ditentukan lebih dahulu prioritasnya

- **Berlaku ketentuan berikut :**
  - Elemen-elemen yang mempunyai prioritas lebih tinggi akan diproses lebih dahulu
  - Dua elemen yang mempunyai prioritas sama akan dikerjakan sesuai dengan urutan pada saat kedua elemen masuk dalam antrian



- Salah satu cara implementasi ***priority queue*** adalah dengan menggunakan *linked list*, ketentuan yang digunakan :
    1. Setiap simpul terdiri dari 3 bagian (informasi, angka prioritas, pointer)
    2. Simpul X mendahului (terletak disebelah kiri) simpul Y, jika prioritas X lebih tinggi dibanding prioritas Y atau jika prioritas X dan Y sama.

Kegunaan *Priority Queue*

- Banyak ditemukan dalam masalah-masalah penjadwalan, misalnya untuk sistem antrian untuk pelayanan pasien dalam ruang praktek dokter, penjadwalan pembayaran tagihan, dsb. Prioritas bisa didefinisikan dalam beberapa aspek. *Priority Queue* dapat digunakan juga untuk pengurutan data dengan key data sebagai harga prioritas : sejumlah data tak terurut dimasukkan ke dalam *Priority Queue* lalu setelah semua masuk satu demi satu data diambil dari *Priority Queue*.
- Contoh lain dalam aplikasi operating system adalah : time sharing system

Penggunaan QUEUE

1. Simulasi antrian di dunia nyata, antara lain:
   - Lalu lintas udara untuk tinggal landas (*take-off*) dan pendaratan (landing)
   - Antrian pembelian tiket di depan loket untuk bis, kereta api, bioskop
   - Antrian mobil di depan gerbang jalan tol
   - Antrian kendaraan di jalan umum

2. Sistem produksi

- Barisan bahan atau komponen yang akan diproses suatu mesin
- Barisan bahan atau komponen yang akan diproses manusia

3. Sistem komputer

- Pemrosesan banyak *job* (tugas) pada sistem *multiprogramming*

Praktik: Stack dan Queue

```python
class Stack:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def push(self, item):
        self.items.append(item)
        print(f"Push: {item}")

    def pop(self):
        if not self.is_empty():
            item = self.items.pop()
            print(f"Pop: {item}")
            return item
        else:
            print("Stack is empty, cannot pop.")
            return None

    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        else:
            return None

    def size(self):
        return len(self.items)

    def display(self):
        print("Stack content:", self.items)
```

```python
# Contoh penggunaan stack
stack = Stack()

stack.push(10)
stack.push(20)
stack.push(30)

stack.display()

print("Peek:", stack.peek())
stack.pop()
stack.display()

stack.pop()
stack.pop()
stack.pop()  # Coba pop saat stack kosong
```

```
Push: 10
Push: 20
Push: 30
Stack content: [10, 20, 30]
Peek: 30
Pop: 30
Stack content: [10, 20]
Pop: 20
Pop: 10
Stack is empty, cannot pop.
```

```python
from collections import deque

# Membuat queue
queue = deque()

# Enqueue
queue.append('X')
queue.append('Y')
queue.append('Z')

print("Queue setelah enqueue:", list(queue))

# Peek
print("Item di depan:", queue[0])

# Dequeue
item = queue.popleft()
print("Item yang dikeluarkan:", item)
print("Queue setelah dequeue:", list(queue))

# Cek kosong dan ukuran
print("Apakah queue kosong?", len(queue) == 0)
print("Ukuran queue:", len(queue))
```

```
Queue setelah enqueue: ['X', 'Y', 'Z']
Item di depan: X
Item yang dikeluarkan: X
Queue setelah dequeue: ['Y', 'Z']
Apakah queue kosong? False
Ukuran queue: 2
```

TERIMA KASIH

Next ---- PERTEMUAN ke-6

TEKNIK ELEKTRO
FTI UJ

ENDANG SRI RAHAYU