Type TQueue3 = <wadah:array[1..10] of character, head:integer, tail:integer >
{Queue model III, kondisi head bisa berputar}
Procedure CreateQueue(output Q:Tqueue3)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan ‘@’, head 0, tail 0}

Function Head(Q:Tqueue3) -> integer
{mengembalikan posisi elemen terdepan}
Function Tail(Q:Tqueue3) -> integer
{mengembalikan posisi elemen terakhir}
Function InfoHead(Q:Tqueue3) -> character
{mengembalikan nilai elemen terdepan}
Function InfoTail(Q:Tqueue3) -> character
{mengembalikan nilai elemen terakhir}
Function isEmptyQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q kosong}
Function isFullQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q penuh}
Function isOneElement(Q:Tqueue3) -> boolean
{mengembalikan true bila Q 1 elemen}

Procedure Enqueue(input/output Q:Tqueue3, input e:character)
{I.S: Q,e terdefinisi, Q mungkin kosong }
{F.S: Q tetap, atau infoTail(Q)=e }
{Proses menambah elemen e ke ekor Q bila belum penuh}
{Bila Tail lama di kapasitas maka Tail baru menjadi 1}

Procedure Dequeue(input/output Q:Tqueue3, output e:character)
{I.S: Q terdefinisi, mungkin kosong }
{F.S: Q tetap, atau e berisi infoHead(Q) lama }
{Proses menghapus elemen e dari head Q bila belum kosong}
{bila Head di kapasitas, maka Head baru menjadi 1}

Procedure PrintQueue(input Q:Tqueue3)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }

Procedure ViewQueue(input Q:Tqueue3)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak kosong Q}

Function SizeQueue(Q:Tqueue3) --> integer
{mengembalikan ukuran antrian}

```
Procedure CreateQueue(output Q:Tqueue3)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan '@', head 0, tail 0}

Kamus Lokal
    i : integer

Algoritma
    i traversal [1..10]
      Q.wadah[i] <-- '@'
    Head(Q) <-- 0
    Tail(Q) <--- 0
```

Function Head(Q:Tqueue3) -> integer
{mengembalikan posisi elemen terdepan}

Kamus Lokal

Algoritma

--> Q.head

Function Tail(Q:Tqueue3) -> integer
{mengembalikan posisi elemen terakhir}

Kamus Lokal

Algoritma

--> Q.tail

```
Function InfoHead(Q:Tqueue3) -> character
{mengembalikan nilai elemen terdepan}

Kamus Lokal

Algoritma
    if (Q.tail =/= 0 AND Q.head =/= 0) then
        -->Q.wadah[Q.head]
    else
        --> '@'
```

cek peluang kondisi Queue

```
Function InfoTail(Q:Tqueue3) -> character
{mengembalikan nilai elemen terakhir}

Kamus Lokal

Algoritma
    if (Q.tail =/= 0 AND Q.head =/= 0) then
        -->Q.wadah[Q.Tail]
    else
        --> '@'
```

```
Function isEmptyQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q kosong}

Kamus Lokal

Algoritma
    if(Q.head = 0 AND Q.tail = 0) then
        --> true
    else
        --> false
```

udah hihi

kalau ngecek semuanya isinya '@' bisa ga?

Function isFullQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q penuh}

Kamus Lokal

Algoritma

if (Q.tail = 10 AND Q.head = 1) then
--> True
else
--> False
kamus lokal

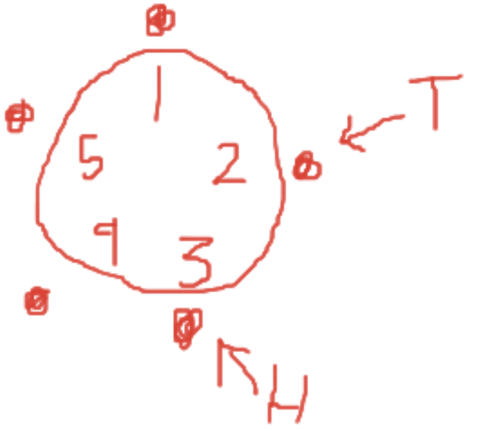
✓clock-based

algoritma
if (Q.head = (Q.tail mod 10 +1) )then
--> true
else
--> false


Kamus Lokal
i : integer

✓reality-based

Algoritma
i <-- 1
while i<=10 do
if Q.wadah[i] =/= '@' then
i <-- i + 1
{endwhile}
if (i <= 10) then
--> False
else
--> True



Function isFullQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q penuh}

ini sama seperti slide sebelumnya Pak

```
Kamus lokal

Algoritma
if (Q.tail > Q.head) then
  if (Q.tail = 10 AND Q. head = 1) then
    --> true
else
  if (Q.tail < Q.head) then
    if (Q.head - Q.tail = 1) then
      --> true
  else
    --> false
```

```
kamus lokal

algoritma
if() then
    --> true
```

```
Function isOneElement(Q:Tqueue3) -> boolean
{mengembalikan true bila Q 1 elemen}

Kamus Lokal

Algoritma
    if(not isEmptyQueue(Q)) then
        if(Q.head = Q.tail) then
            --> true
        else
            --> false
```

```
Procedure Enqueue(input/output Q:Tqueue3, input e:character)
{I.S: Q,e terdefinisi, Q mungkin kosong }
{F.S: Q tetap, atau infoTail(Q)=e }

 Kamus Lokal

 Algoritma
    if(not isFullQueue(Q))then
        if(isEmptyQueue(Q)) then
            Head(Q) <-- 1
        Tail(Q) <-- (Tail(Q) + 1) mod 10
        Q.wadah[Tail(Q)] <-- e
```

Procedure Dequeue(input/output Q:Tqueue3, output e:character)
{I.S: Q terdefinisi, mungkin kosong }
{F.S: Q tetap, atau e berisi infoHead(Q) lama }
{Proses menghapus elemen e dari head Q bila belum kosong}
{bila Head di kapasitas, maka Head baru menjadi 1}

kamus lokal

```
algoritma
if(not isEmptyQueue(Q)) then
    if (isOneElement(Q)) then
        e <-- Q.wadah[head(Q)]
        Q.wadah[head(Q)] <-- '@'
        head(Q) <-- 0
        tail(Q) <-- 0
    else {kasus >1 elemen dan kasus penuh}
        e <-- Q.wadah[head(Q)]
        q.wadah[head(Q)] <-- '@'
        head(Q) <-- (head(Q)+1) mod 10
else
    e <-- '@'
```

cek
1. kasus kosong
2. kasus penuh
3. kasus 1 elemen
4. kasus >1 elemen

```
if(not isEmptyQueue(Q)) then
    e <-- Q.wadah[head(Q)]
    Q.wadah[head(Q)] <-- '@'
    if (isOneElement(Q)) then
        head(Q) <-- 0
        tail(Q) <-- 0
    else {lebih dari 1 elemen}
        head(Q) <-- (head(Q)+1) mod 10
else {elemen kosong}
    e <-- '@'
```

```
Procedure PrintQueue(input Q:Tqueue3)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }

kamus lokal
    i : integer

algoritma
if (not isEmptyQueue(Q)) then
     i traversal 1.. 10
          Output(Q.wadah[i])
     {end traversal}
```

bukan fungsi !!!

mengembalikan nilai return

Procedure ViewQueue(input Q:Tqueue3)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak kosong Q}

kamus lokal
    i : integer

algoritma
if (not isEmptyQueue(Q)) then
    i traversal 1.. tail(Q)
        if (Q.wadah[i] =/= '@') then
            --> Q.wadah[i]
    {end traversal}


Kamus Lokal
i: integer

Algoritma
if (not isEmptyQueue(Q)) then
    if (head(Q) < tail(Q)) then
        i traversal head(Q)..tail(Q)
            output (Q.wadah[i])
    else if (head(Q) > tail(Q)) then
        i traversal head(Q)..tail(Q)+10
            if (i>10) then
                output(Q.wadah[i mod 10])
            else
                output(Q.wadah[i])
    else {one element}
        output(Q.wadah[(head(Q)])

if (not isEmptyQueue(Q)) then
    if (head(Q) < tail(Q)) then
        i traversal head(Q)..tail(Q)
            output(Q.wadah[i])
    else
        if (head(Q) > tail(Q)) then
            i traversal head(Q)..5
                output(Q.wadah[i])
            i traversal 1..tail(Q)
                output(Q.wadah[i])
        else {one element}
            output(Q.wadah[(head(Q)])

kalo gini gimana pak?

```
Function SizeQueue(Q:Tqueue3) --> integer
{mengembalikan ukuran antrian}

Kamus Lokal
    ukuran : integer

Algoritma
    if(Q.tail >= Q.head) then
        ukuran <-- (Q.tail - Q.head) + 1
    else
        ukuran <-- 10 - (Q.head - Q.tail) + 1
```

```
Kamus Lokal

Algoritma
    if(Q.tail >= Q.head) then
        --> (Q.tail - Q.head) + 1
    else
        --> 10 - (Q.head - Q.tail) + 1
```