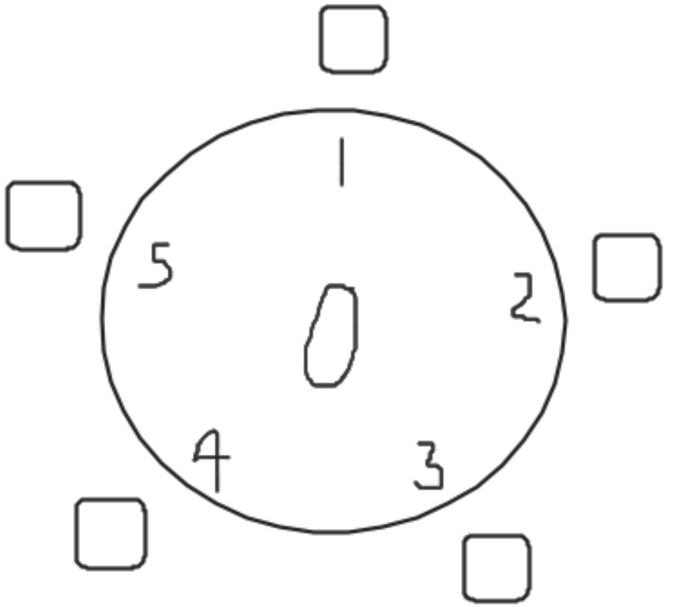
```
Type TQueue3 = <
                    wadah:array[1..10] of character,
                    head:integer,
                    tail:integer >
{Queue model III, kondisi head bisa berputar}
```



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

Kamus lokal
    i : integer { iterator }

Algortima
  Q.head <-- 0
  Q.tail  <-- 0
  i travesal 1 .. 10
    Q.wadah[i] <-- '@'
```

```
Kamus Lokal
    i: integer

Algoritma
    Q.head <- 0
    Q.tail <- 0
    i tarversal 1..10
      Q.wadah[i] <- '@'
```

```
Kamus lokal
    i : integer

Algoritma
i traversal 1 .. 10
    Q.wadah[i] <- '@'

Q.head <- 0
Q.tail <- 0
```

Sama aja

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

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


kamus lokal

algoritma
    if not isEmptyQueue(Q) then
          --> Q.wadah[Q.tail]
    else
         --> '@'
```

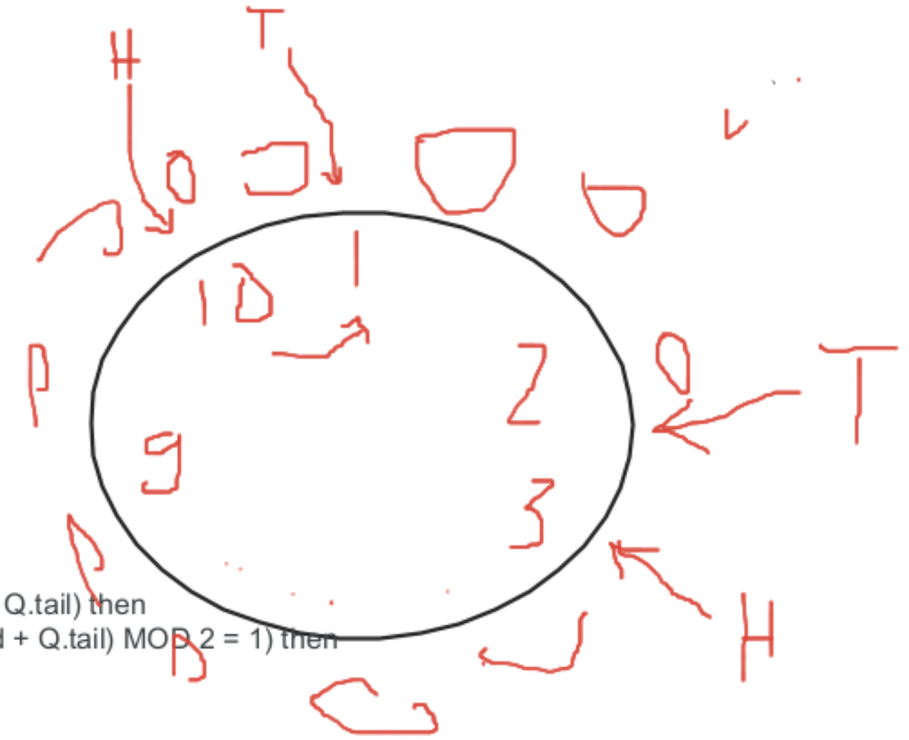
```
Function isEmptyQueue(Q:Tqueue3) -> boolean
{mengembalikan true bila Q kosong}

  Kamus Lokal

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

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

Kamus Lokal

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

```
kamus lokal
   i : integer
   cek : boolean

Algortima
   cek <- true
   i <- 1
   while cek = true AND i <= 10 do
   if Q.wadah[i] = '@' then
      cek <- false
       i <- i + 1

   -> cek
```



```
if (Q.head > Q.tail) then
  if ((Q.head + Q.tail) MOD 2 = 1) then
   --> true
  else
   --> false
else
  if (Q.head = 1 AND Q.tail = 10) then
   --> true
  else
   --> false
```

```
Kamus lokal
   i : integer
   cek : integer

Algoritma
   cek <- 0
   i traversal 1 .. 10
      if (Q.wadah[i] = '@') then
         cek <- cek + 1
   if (cek = 0) then
      -> true
   else
      -> false
```

```
Kamus lokal
   i : integer
   cek : boolean

Algoritma
   cek <- true
   i traversal 1 .. 10
      if Q.wadah[i] = '@' then
      cek <- false

   -> cek
```

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

modulo 12
bukan pak?

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

Kamus Lokal

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

bener gini kan ya..? || iya :3

iyakk

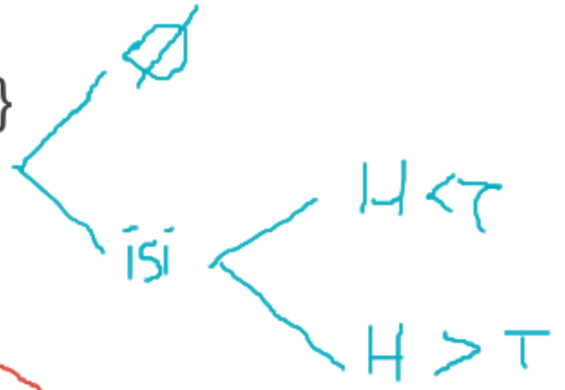
tp kalau head dan tailnya ga di 1 gimana? Misal abis di dequeue

Kamus lokal

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

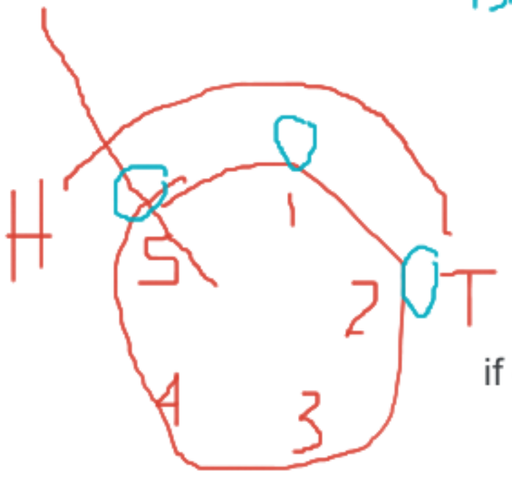
```
Procedure PrintQueue(input Q:Tqueue3)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }

Kamus Lokal
    i : integer

Algoritma
    i traversal 1..10
       output(Q.wadah[i])
```

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
    if (head(Q) < tail(Q) then
      i traversal [Q.head .. Q.tail]
        output(Q. wadah[i])
    else
      i traversal [Q.head .. 5]
        output(Q. wadah[i])
      i traversal [1 .. Q.tail]
        output(Q. wadah[i])



pikirkan bila tail < head

```
if (not isEmptyQueue(Q)) then
    i traversal[Q.head..Q.head+sizeQueue(Q)-1]
        output Q.wadah[(i-1 mod 5) + 1]
```

```
if(not isEmptyQueue(Q))then
  i traversal[Q.head...Q.head+sizeQueue(Q)-1]
    if (i = 5) then
        output Q.wadah[5]
    else
        output Q.wadah[i mod 5]
```

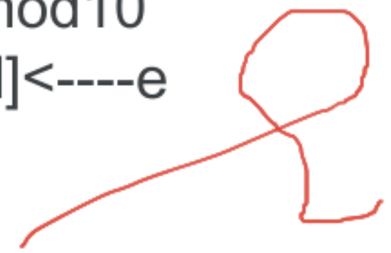
kalau pake i sampai tail +5
nanti i nya di mod 5 lagi

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}

Kamus Lokal

Algoritma

```
if(not isFullQueue(Q))then

    if(isEmptyQueue(Q))then
        Q.head<---1

    tail<---(tail+1)mod10
    Q.wadah[Q.tail]<----e
```



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

```
Kamus Lokal

Algoritma
 if(isEmptyQueue(Q) then
    --> 0
 else
    --> Q.tail - Q.head + 1
```

eh kalo tail itu udah jadi ukuran?

katanya bukan gini tadi

```
Kamus Lokal

Algoritma
 if(isEmptyQueue(Q)) then
    --> 0
 else
    if Head(Q) > Tail(Q) then
       --> 10 - Head(Q) + Tail(Q) + 1
    else {Head(Q) < Tail(Q)}
       -->  Tail(Q) - Head(Q) + 1
```

```
kamus lokal
   int i
   int j
   int temp

algoritma :
   if (isEmptyQueue(Q)) then
      -->0
else
    j = 1
    i traversal Q.head .. Q.tail
        Q.head <-- j
        j = j + 1
        temp <-- j
    --> temp
```

Kamus lokal

```
Algoritma
   if (isEmptyQueue(Q)) then
      --> 0
   else
      --> (10 - Head(Q) + Tail(Q)) mod 10 + 1
```

head bisa lebih besar dari pada tail

oh gitu makasih

tapi tambahin kondisi kosong biar kalo kosong jadi 0

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 {queue tidak kosong}
    e <- Q.wadah[Q.head]
        Q.wadah[Q.head] <- '@'
    if (Q.head = Q.tail) then {1 elemen saja, queue jadi kosong}
      Q.head <- 0
      Q.tail <- 0
    else {jumlah elemen lebih dari 1}
            Q.head <- (Q.head mod 10) + 1
  else {queue kosong}
    e <- '@'
```

```
Kamus lokal

Algoritma
  if not(isEmptyQueue(Q)) then {queue tidak kosong}
    e <- Q.wadah[Q.head]
        Q.wadah[Q.head] <- '@'
    if (Q.head = Q.tail) then {1 elemen saja, queue jadi kosong}
      Q.head <- 0
      Q.tail <- 0
    else {jumlah elemen lebih dari 1}
            if (Q.head = 10) then {head di akhir, putar ke 1}
              Q.head <- 1
            else {head tidak di akhir}
              Q.head <- Q.head + 1

  else {queue kosong}
    e <- '@'
```