```
Type TQueue = <wadah:array[1..10] of character,
              head:integer,
              tail:integer >
{Queue model I, kondisi head 0 atau 1}
{pergeseran maju pada elemen ketika dequeue}
Procedure CreateQueue(output Q:TQueue)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan ‘ ’,
head 0, tail 0}
Function Head(Q:TQueue) -> integer
{mengembalikan posisi elemen terdepan}
Function Tail(Q:TQueue) -> integer
{mengembalikan posisi elemen terakhir}
Function InfoHead(Q:TQueue) -> character
{mengembalikan nilai elemen terdepan}
Function InfoTail(Q:TQueue) -> character
{mengembalikan nilai elemen terakhir}
Function isEmptyQueue(Q:TQueue) -> boolean
{mengembalikan true bila Q kosong}
Function isFullQueue(Q:TQueue) -> boolean
{mengembalikan true bila Q penuh}
```

```
Procedure Enqueue(input/output Q:TQueue, 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}
Procedure Dequeue(input/output Q:TQueue, 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}
{lalu geser maju 1 langkah semua elemen di
belakang head}
Procedure PrintQueue(input Q:TQueue)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q
Procedure ViewQueue(input Q:TQueue)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak
kosong Q}
Function sizeQueue(Q:TQueue) -> integer
{mengembalikan panjang/banyak elemen}
```

```
Procedure CreateQueue(output Q:TQueue)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan ‘ ’,
head 0, tail 0}
Kamus Lokal
      i : integer {iterator}
Algoritma
     Q.head <- 0
     Q.tail <- 0
     i traversal [1..10]
          Q.wadah[i] <- ' '
```

Function Head(Q:TQueue) -> integer
{mengembalikan posisi elemen terdepan}
Kamus lokal

Algoritma
    -> Q.head

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

Kamus Lokal

Algoritma
    -> Q.tail

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

Algoritma
    if(Head(Q) = 1) then { kasus Q berisi }
      -> Q.wadah[Head(Q)]
   else
     -> ' '     { kasus Q kosong }
```

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

Algoritma
    if (Tail(Q)>0) then { kasus Q berisi }
      -> Q.wadah[Tail(Q)]
    else        { kasus Q kosong }
      -> ' '
```

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

Kamus Lokal

Algoritma
        if Head(Q)= 0 then {apabila queue kosong} AND tail(Q)=0
            -> true
        else {apabila queu tidak kosong}
           -> false
```

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

kamus lokal

algoritma
        if Tail(Q) = 10 then {apabila Q penuh}    AND head(Q)=1
           -> true
        else {Q tidak penuh}
          -> false
```

```
Procedure Enqueue(input/output Q:TQueue, 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}
kamus lokal

algoritma
        if not isFullQueue(Q) then { menghilangkan case queue penuh }
                if isEmptyQueue(Q) then { antrean baru terisi head akan pindah ke 1}
                        Q.head <- 1
                        Q.tail <- Q.tail + 1
                        Q.wadah[Q.tail] <- e
                else
                        Q.tail <- Q.tail + 1
                        Q.wadah[Q.tail] <- e
```

{atau}

```
        if isEmptyQueue(Q) then
                Q.head <- 1
                Q.tail <- 1
                Q.wadah[Q.tail] <- e
        else
                if not isFullQueue(Q) then
                        Q.tail <- Q.tail + 1
                        Q.wadah[Q.tail] <- e}
```

```
Procedure Enqueue(input/output Q:TQueue, 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}
kamus lokal

algoritma
        if not isFullQueue(Q) then { menghilangkan case queue penuh }
                if isEmptyQueue(Q) then { antrean baru terisi head akan pindah ke 1}
                        Q.head <- 1

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

```
Procedure Dequeue(input/output Q:TQueue, 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}
{lalu geser maju 1 langkah semua elemen di belakang head}
kamus lokal
        i : integer { iterator saat geser }

algoritma
        if not isEmptyQueue(Q) then { menghilangkan case queue kosong }
                e <- InfoHead(Q)

                i traversal [Head(Q) .. Tail(Q) - 1] { menggeser antrean }
                        Q.wadah[i] <- Q.wadah[i+1]
                Q.wadah[Tail(Q)] <- ' '

                if Q.tail = 1 then { antrean tinggal 1, head akan pindah ke 0}
                        Q.head <- 0
                Q.tail <- Q.tail - 1
        else {kasus kosong}
                e <- ' '
```

```
Q.tail <- Q.tail - 1
if Q.tail = 0 then
    Q.head <- 0
```

```
Procedure PrintQueue(input Q:TQueue)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }
Kamus Lokal
      i : integer {iterator}
Algoritma
      i traversal [1..10]
          output Q.wadah[i]
```

```
Procedure ViewQueue(input Q:TQueue)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak kosong Q}
Kamus Lokal
    i : integer {iterator}
Algoritma
    if (not isEmptyQueue(Q)) then {apabila tidak kosong}
        i traversal [1..Tail(Q)]
            output Q.wadah[i]
    else {apabila kosong}
        output 'Queue kosong'
```

alternatif:

```
if (not isEmptyQueue(Q)) then {apabila tidak kosong}
    i traversal [1..Q.tail]
        output Q.wadah[i]
else {apabila kosong}
    output 'Queue kosong'
```

Function sizeQueue(Q:TQueue) -> integer
{mengembalikan panjang/banyak elemen}

Kamus Lokal

Algoritma
    -> Q.tail

program utama