DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
Type TQueue = <wadah:array[1..10] of character, head:integer, tail:integer >
{Queue model II, kondisi head bergeser}
{kembali ke 1 bila tail mencapai kapasistas}

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}

DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
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

  algoritma
  i traversal 1.. 10
       Q.wadah[i] = ' '
  Q.head = 0
  Q.tail = 0
```

true/false?

```
head(Q) <-- 0
tail(Q) <-- 0
```

yang bener

```
kamus lokal
i: integer

algoritma
i traversal 1.. 10
     Q.wadah[i] <-- ' '
Q.head <-- 0
Q.tail <-- 0
```

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

Kamus Lokal

Algoritma

--> Q.head

fungsi dalam C
int head(tqueue Q)

makro
#define head(Q)  (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}
{Asumsi : bila Q kosong, Q.wadah=#}
    Kamus Lokal

bocor bila Q kosong

Algoritma
if(Q.head=/=0 and Q.tail=/=0) then
    -->Q.wadah[Q.head]
else
    -->'#'

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

bocor bila Q kosong

Kamus lokal

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

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

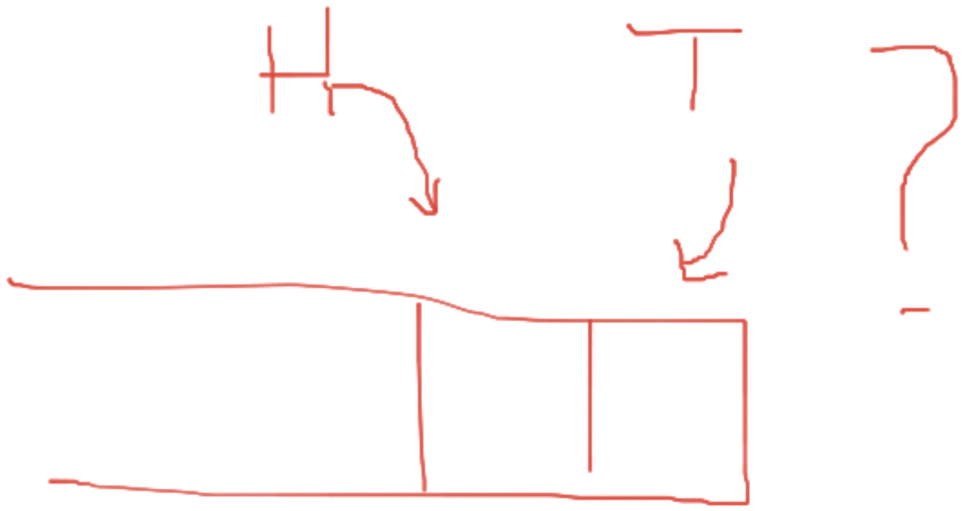
Kamus Lokal

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

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

Kamus Lokal

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



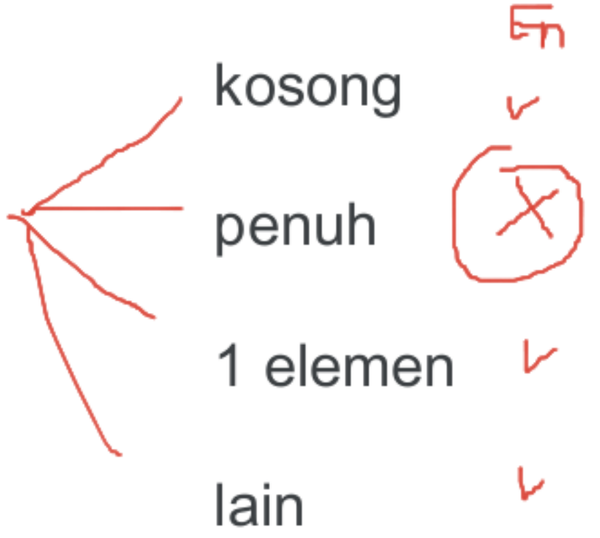
kamus lokal

algoritma
if(Q.tail = 10) then
    -->true

Kamus Lokal

algoritma
--> Q.tail = 10

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}
{bila tail di kapasitas, head direset 1 diikuti elemen lain}




kamus lokal

algoritma
```
if(not isFullQueue(Q)) then
    if(isTailStop(Q)) then
        ResetHead(Q)
    Tail(Q) <-- Tail(Q) + 1
    Q.wadah[Tail(Q)] <-- e
```

kamus lokal
i: integer
Algoritma
```
if (isEmptyQueue(Q) or not isFullQueue(Q)) then
    if(isTailStop(Q)) then
        ResetHead(Q)
    tail(Q) <-- tail(Q) +1
    Q.wadah[tail(Q)] = e
```


baik pak terima kasih

```
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 }

Kamus Lokal
    i : integer

Algoritma
    if(not isEmptyQueue(Q)) then
        e <-- infoHead(Q)
        infoHead(Q) < -- ' '
        if(Head(Q) = Tail(Q)) then
            Head(Q) <-- 0
            Tail(Q) <-- 0
        else
            Head(Q) <-- Head(Q) + 1
```

Dah salah nih

```
is (not isEmptyQueue(Q)) then
    e <-- infoHead(Q)
```

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

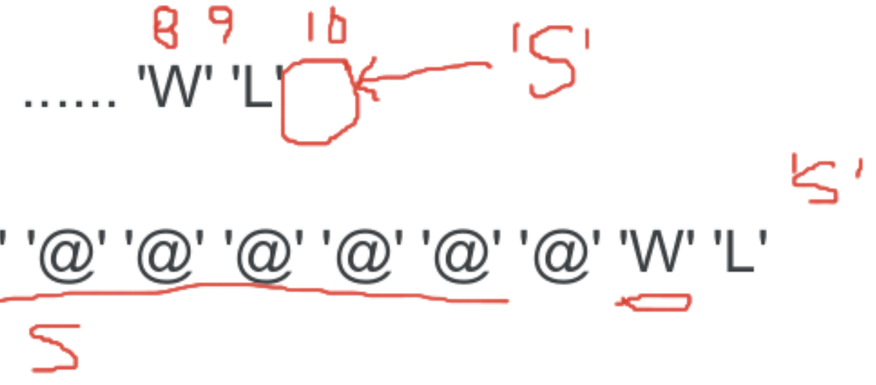
```
kamus lokal
i: integer

algoritma
if (not isEmptyQueue(Q)) then
    i treversal 1.. tail(Q)
        output (Q.wadah[i])
        i <-- i+1
    {end traversal}
```

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

kamus lokal

Algoritma
--> Tail(Q) - Head(Q) + 1

Function IsTailStop(Q:TQueue2) -> boolean
{mengembalikan true jika Tail tidak dapat lagi geser}
{karena sudah di posisi kapasitas}




kamus lokal

algoritma
--> tail(Q) = 10

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

--> tail(Q) = 10 AND head(Q) = 9

```
Procedure ResetHead(input/output Q:TQueue2)
{I.S:Tail=kapasitas, head>1; F.S:head=1;
{Proses: mengembalikan Head ke indeks 1 }
{Elemen selain head ikut bergeser menyesuaikan}
  kamus lokal
     i : integer

  algoritma
     if(isTailStop(Q)) then
        i traversal [1..(sizeQueue(Q))]
           Q.wadah[i] <-- Q.wadah[Q.head + i - 1]
           Q.wadah[Q.head + i - 1] <-- ' '
        Head(Q) <-- 1
        Tail(Q) <-- i
```

```
kamus lokal
     temp: integer
      i: integer
     j: integer

algoritma
   if (isTailStop(Q)) then
        temp <- 1
        i traversal head(Q)..tail(Q)
              Q.wadah[temp] <- Q.wadah[i]
              temp <-- temp + 1

        j traversal (tail(Q)+1) .. 10
              Q.wadah[j] <-- '@'
        head(Q) <-- 1
        tail(Q) <-- temp
```