DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
Type TQueue2 = <wadah:array[1..10] of character, head:integer, tail:integer >
{Queue model II, kondisi head bisa geser 0..kapasitas}
{pergeseran maju pada elemen ketika Tail terhenti}

Procedure CreateQueue(output Q:TQueue2)
{I.S: - ; F.S: Q terdefinisi}
{Proses: mengisi elemen wadah dengan ‘ ’, head 0, tail 0}
Function Head(Q:TQueue2) -> integer
{mengembalikan posisi elemen terdepan}
Function Tail(Q:TQueue2) -> integer
{mengembalikan posisi elemen terakhir}
Function InfoHead(Q:TQueue2) -> character
{mengembalikan nilai elemen terdepan}
Function InfoTail(Q:TQueue2) -> character
{mengembalikan nilai elemen terakhir}
Function isEmptyQueue(Q:TQueue2) -> boolean
{mengembalikan true bila Q kosong}
Function isFullQueue(Q:TQueue2) -> boolean
{mengembalikan true bila Q penuh}

DEKLARASI/DEFINISI&SPESIFIKASI TIPE & PROTOTIPE
Function IsTailStop(Q:TQueue2) -> boolean
{mengembalikan true jika Tail tidak dapat lagi geser}
{karena sudah di posisi kapasitas}
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}
Procedure Enqueue(input/output Q:TQueue2, 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 semua elemen lain}
Procedure Dequeue(input/output Q:TQueue2, 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}
Procedure PrintQueue(input Q:TQueue2)
{I.S:-; F.S:-; Proses: menampilkan kondisi wadah Q }
Procedure ViewQueue(input Q:TQueue2)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak kosong Q}

Procedure CreateQueue(output Q:TQueue2)
{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:TQueue2) -> integer
{mengembalikan posisi elemen terdepan}

kamus lokal

Algoritma
--> Q.Head

int Head(tqueue2 Q)

#define head(Q) (Q).head

Function Tail(Q:TQueue2) -> integer
{mengembalikan posisi elemen terakhir}

Kamus lokal

Algoritma
--> Q.tail

Function isEmptyQueue(Q:TQueue2) -> boolean
{mengembalikan true bila Q kosong}

```
kamus lokal
    cek: boolean

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

Function isFullQueue(Q:TQueue2) -> boolean
{mengembalikan true bila Q penuh}

```
kamus lokal
   cek:boolean

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

Function InfoHead(Q:TQueue2) -> character
{mengembalikan nilai elemen terdepan}

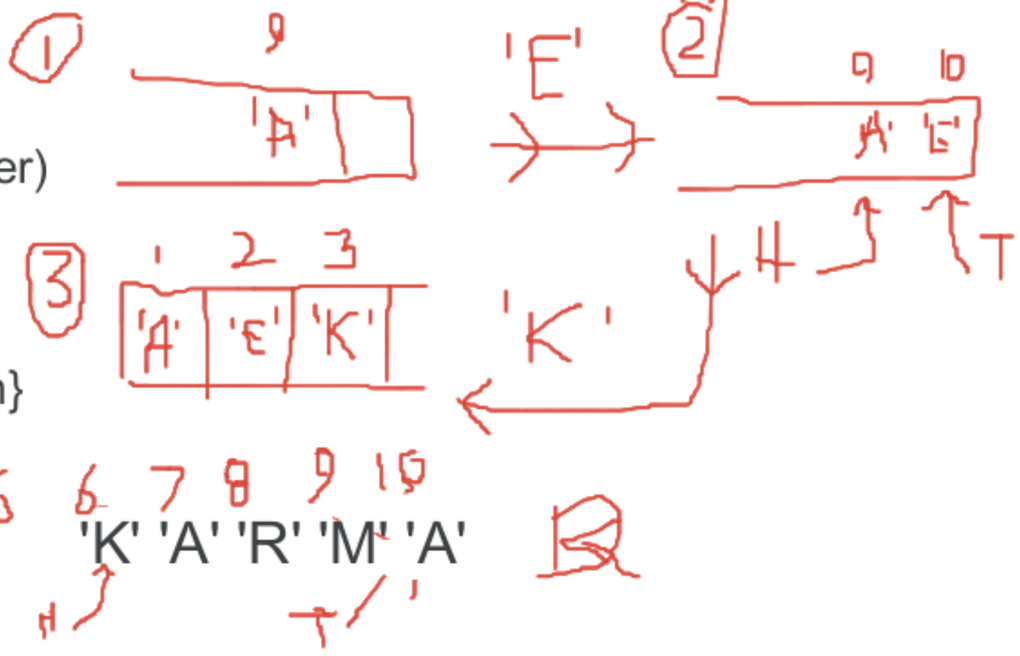
```
kamus lokal

Algoritma
  if not isEmptyQueue(Q) then
     -->Q.wadah[Head(Q)]
  Else {Q kosong}
     ---> ' '
```

Function InfoTail(Q:TQueue2) -> character
{mengembalikan nilai elemen terakhir}

```
Kamus lokal

Algoritma
   if not isEmptyQueue(Q) then
      --> Q.wadah[Tail(Q)]
  Else { Q kosong}
    --> ' '
```



```
Procedure Enqueue(input/output Q:TQueue2, 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 semua elemen lain}
Kamus lokal
    i,j : integer              {iterator}

Algoritma
        if not(isTailStop(Q)) then {Tail tidak di akhir, enqueue biasa}
           Q.tail <- Q.tail + 1
           Q.wadah[Q.tail] <- e
           if (Q.tail = 1) then {kondisi awal kosong}
             Q.head <- 1
        else {Tail di akhir, reset head / queue tetap}
            if not(isFullQueue(Q)) then {Tidak penuh, reset head}
                      {bisa menggunakan resetHead(Q) terlebih dahulu}
                i traversal [Q.head..Q.tail]
                  Q.wadah[i-Q.head+1] <- Q.wadah[Q.head]
                       Q.tail <- i - Q.head + 1
                       Q.head <- 1
               Q.wadah[Q.tail] <- e
                       j traversal [Q.tail+1..10] {Reset elemen setelah tail
                            Q.wadah[j] <- ' '
```


```
Procedure Dequeue(input/output Q:TQueue2, 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}

Kamus Lokal
  i:integer{iterasi}
Algoritma
  if(not isEmptyQueue(Q)then
     e<---Q.wadah[Q.head]
     Q.wadah[Q.head]<---- ' '
     if(Q.head!=10)then
        Q.head<---Q.head+1
     else
        Q.head<--- 0{karena Q menjadi kosong}
else
    e<--' '
```

KASUS KOSONG?
KASUS PENUH?
KASUS 1 ELEMEN?
KASUS LAIN

```
Procedure PrintQueue(input Q:TQueue2)
{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:TQueue2)
{I.S:-; F.S:-; Proses: menampilkan info elemen tak kosong Q}

Kamus lokal
   i : integer

Algoritma
   if(not isEmptyQueue(Q)) then
     i traversal [Q.head .. Q.tail]
         output(Q.wadah[i])
```

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

1	2	3	4	5	6	7	8	9	10
'M'	'E'	'N'	'T'	'O'	'K'	'N'	'I'	'H'	

Kamus Lokal

Algoritma

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

Kamus Lokal
cek : boolean

algoritma
if (not isFullQueue(Q)
AND Q.Tail = 10) then
cek <- true
else
cek <- false

-> cek

```
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
      j :integer

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

         i traversal Q.tail+1 .. 10
              Q.wadah[i] <- ' '
```