```
procedure initTabel( output T: Tabel) {definisi}
{I.S: -}
{F.S: T terdefinisi }
{proses: mengisi elemen dengan -999, size=0}
-----------------------------------------------------------------
kamus lokal
  i : integer {iterator}
-----------------------------
algoritma
  T.size <-- 0
  i traversal 1..10
     T.wadah[ i ] <--     -999
  {end traversal T terdefinisi}
```

function getNewTable() --> Tabel
procedure getNewTable(output T: Tabel, input N : Integer)
{I.S : T Terdefinisi}
{F.S : T terisi dengan elemen -999}
Kamus Lokal
N : Integer
Algoritma
T.size <-- 0
N<-- -999
{mengembalikan tabel dengan size=0, setiap elemen wadah=-999}

```
function getNewTable() --> Tabel
   {mengembalikan tabel dengan size=0, setiap elemen wadah=-999}
kamus lokal
   T:Tabel
   i: integer
algoritma
   T.size <-- 0
   i traversal 1..10
      T.waddah[ i ] <--  -999
   --> T
```

```
aplikasi:
kamus
   A:Tabel
algoritma
   A <-- getNewTable()
```

```
module Tabel
procedure printTable ( input T:Tabel )
      {I.S.: T terdefinisi}
      {F.S.: -}
      {Proses: menampilkan semua elemen T ke layar}

body Table
procedure printTable ( input T:Tabel )
      {I.S.: T terdefinisi}
      {F.S.: -}
      {Proses: menampilkan semua elemen T ke layar}

Kamus lokal
      i : integer
      T : integer

Algoritma
    T.size <-- T
    i traversal 1 .. T.size
    output T.wadah[ i ]
    printf(T)
```

i traversal 1..10
    output T.wadah[i]

```
module Tabel
procedure viewTable ( input T:Tabel )
      {I.S.: T terdefinisi}
      {F.S.: -}
      {Proses: menampilkan elemen T yang terisi ke layar}

body Table
procedure viewTable ( input T:Tabel )
      {I.S.: T terdefinisi}
      {F.S.: -}
      {Proses: menampilkan elemen T yang terisi ke layar}

Kamus lokal
      i : integer
      T : integer

Algoritma
    T.size <-- T
    i traversal 1 .. T.size
      input T
      output
      T.wadah[i] <-- T
printf(T)
```

i traversal 1..T.size
    output T.wadah[i]

```
function isEmptyTable( T: Tabel) -> boolean
    {mengembalikan True jika T kosong }
kamus lokal
    i: integer {iterator}
    empty: boolean
algoritma
    empty <-- True
    i <-- 1
    while i <= T.size Do
        if T.wadah[i] =/= -999 then
            empty <-- False
        i <--- i + 1
    {end while}
    --> empty
```

{terisi}

```
function countEmpty( T: Tabel) -> integer
    {mengembalikan banyak elemen T yang belum terisi }
kamus lokal
    i: integer {iterator}
    count: integer
algoritma
    i <-- 1
    count <-- 0
    while i <= T.size Do
        if T.wadah[i] =/= -999l then
            count <-- count + 1
        i <-- i + 1
    {end while}
--> count
```

size

wadah[i]

algoritma

--> (T.size = 0)

```
function isFullTable( T: Tabel) -> boolean
  {mengembalikan True jika T penuh }

 kamus lokal
   i: integer
   full: boolean
 algoritma
   full <-- True
   while (full) do
     i traversal 1.. T.size
      if (T.wadah[i] != 0) then
          full <-- True
       else
        full <-- False
   --> full
algoritma:
  if (T.size = 10) then     --> T.size=10
     --> true
else
    --> false
```

```
function getSize( T: Tabel) -> integer
  {mengembalikan banyak elemen pengisi T }

 kamus lokal
   i: integer
   x:integer
 algoritma
   x <-- 0
   i traversal 1.. T.size
     if (T.wadah[i] != 0) then
          x <- x +1
    --> x


Atau
kamus lokal

 algoritma
   --> T.Size
```

```
procedure searchX1 ( input T:Tabel, input X: integer, output Pos: integer )
  {I.S.: T terdefinisi, X terdefinisi }
  {F.S.: Pos berisi posisi pertama ketemu X di T.wadah, atau -999 jika tidak ketemu }
  {Proses: mencari elemen bernilai X dalam T.wadah}
```


```
kamus lokal
  i: integer
algoritma
  i traversal 1.. T.Size
    if (T.wadah[i] = X) then
      Pos <- i
      break
```

```
Pos <--  -999
i traversal 1..T.size
  if (T.wadah[ i ] = X ) then
      Pos <-- i
```

```
if isEmptyTable(T)=true  then
    Pos <--  -999
else {T.size>0}
    i <-- 1
    while (T.wadah[i]=/=X) and (i<T.size) do
          i <-- i+1
    {T.wadah[i]=X OR i=T.size}
    if (T.wadah[i]=X) then
        Pos <--  i
    else
        Pos <--  -999
```

Pos

1,1,2,4,3,2,5,1,1,5

```
iterate
    i <-- i+1
stop (T.wadah[i]=X OR i=T.size)
```

```
aplikasi
  A: Tabel
  posisi : integer
algoritma
   posisi <-- 90
   A <-- getNewTable()
   searchX1 ( A, 2, posisi
   output posisi {-999}
```

```
procedure countX ( input T:Tabel, input X: integer, output Byk: integer )
    {I.S.: T terdefinisi, X terdefinisi }
    {F.S.: Byk berisi banyaknya X di T.wadah, atau 0 jika tidak ketemu }
    {Proses: menghitung elemen bernilai X dalam T.wadah}


    kamus lokal
      i: integer
    algoritma
      Byk <- 0
      i traversal 1.. T.Size
        if (T.wadah[i] = X) then
          Byk <- Byk + 1
```

```
function getMaxElm ( T: Tabel) -> integer
  {mengembalikan nilai elemen terbesar }
    {ASUMSI, bila tabel kosong, max=???}
  kamus lokal
    i: integer
    temp: integer
    max:integer
  algoritma
    if NOT isEmptyTable(T) then
    temp <- T.wadah[1]
    i traversal 1.. T.Size
      if (temp > T.wadah[i + 1]) then
        max <- temp
      else
        temp <- T.wadah[i + 1]

   -> max
```

```
function getMinElm ( T: Tabel) -> integer
    {mengembalikan nilai elemen terkecil, tapi bukan -999 }

  kamus lokal
    i: integer
    temp: integer
    min:integer
  algoritma
    temp <- T.wadah[1]
    i traversal 1.. T.Size
      if (T.wadah[i+1] =/= -999) then
        if (temp < T.wadah[i + 1]) then
          min <- temp
        else
          temp <- T.wadah[i + 1]

    -> min
```

```
procedure addX ( input/output T:Tabel, input X: integer )
    {I.S.: T terdefinisi, X terdefinisi }
    {F.S.: isi T.wadah bertambah 1 elemen jika belum penuh}
    {Proses: mengisi elemen T.wadah dengan nilai X}
```

```
kamus lokal
  i: integer
algoritma
  i traversal 1.. T.Size
    if (T.wadah[i] = -999) then
      input(X)
      T.wadah[i] <- x
```

```
if T.size<10 then {=NOT isFullTabel(T)}
   T.size <-- T.size + 1
   T.wadah[ T.size ] <-- X
```

1,1,2,5,..... size = 4
size 5
wadah[5] <- 8

```
aplikasi:
kamus
 A: Tabel
  r : integer
algoritma
  initTabel(A)
  populateTabel(A,5)
  input r {8}
  addX(A,r)
```

```
procedure delX ( input/output T:Tabel, input X: integer )
    {I.S.: T terdefinisi}
    {F.S.: isi T.wadah berkurang 1 elemen jika belum kosong}
    {Proses: menghapus 1 elemen bernilai X, geser semua elemen sisa}

kamus lokal
  i: integer          ;posisi:integer
  temp:integer
algoritma
  while (T.wadah[i] =/= X)
    i <- i+1
  if (T.wadah[i] = X) then
      temp <- i
   i traversal temp.. T.Size
      T.wadah[temp] <- T.wadah[temp+1]
```

```
if NOT isEmptyTabel(T) then
    searchX1(T,X,posisi)
    if posisi =/= -999 then
        i traversal posisi..T.size
            T.wadah[i] <-- T.wadah[i+1]
        T.size <-- T.size - 1
```

```
if NOT isEmptyTabel(T) then
    i <-- 1
    {cari X}
    while (T.wadah[ i ] =/= X) AND (i<T.size) do
        i <-- i+1
    {T.wadah[ i ] = X OR i<T.size }
    if (T.wadah[ i ] = X) then {ketemu}
        {geser}
        while i<T.size do
            T.wadah[i] <- T.wadah[i+1]
        {i=T.size}
        T.size <-- T.size - 1
```

procedure delAllX (input/output T:Tabel, input X: integer)
{I.S.: T terdefinisi}
{F.S.: isi T.wadah berkurang semua elemen bernilai X jika belum kosong}
{Proses: menghapus semua elemen bernilai X, geser elemen sisa}

lakukan delX sebanyak countX

```
procedure updateX ( input/output T:Tabel, input X: integer, input Y: integer )
    {I.S.: T terdefinisi, Y >= 0}
    {F.S.: isi T.wadah bernilai X diganti nilainya dengan Y}
    {Proses: mengganti elemen pertama bernilai X dengan nilai Y }
```