Latihan Soal
ADT List dengan Array

Latihan Soal 1

Asumsikan bahwa sudah didefinisikan list memanfaatkan array statik yang direpresentasikan secara eksplisit dalam notasi algoritmik (alt-2 rata kiri).

Buatlah masing-masing fungsi/prosedur berikut ini untuk representasi struktur data array tersebut dalam notasi algoritmik.

Gunakan sedapat mungkin primitif-primitif lain yang tersedia.

Latihan Soal 1

```
function isSimetris (l: List) → boolean
{ Menghasilkan true jika List l simetrik. }
{ List disebut simetrik jika:
  - elemen pertama = elemen terakhir,
  - elemen kedua = elemen sebelum terakhir,
  - dan seterusnya.
  List kosong adalah List simetris }

function plusTab (l1,l2: List) → List
{ Prekondisi: l1 dan l2 berukuran sama dan tidak kosong. }
{ Mengirimkan l1+l2, yaitu penjumlahan setiap elemen l1 dan l2
  pada indeks yang sama (seperti penjumlahan vektor dalam matematika). }

function countOccurence(l: List, x: ElType) → integer
{ Menghasilkan berapa banyak kemunculan elemen bernilai x di List l. }
{ Jika l kosong, menghasilkan 0. }
```

Latihan Soal 1

```
function isEqual (l1,l2:List) → boolean
{ Mengirimkan true jika l1 setara dengan l2, yaitu jika ukuran
  l1 sama dengan ukuran l2 dan semua elemen l1 dan l2 pada indeks
  yang sama bernilai sama; L1 dan L2 tidak kosong. }

function indexOf (l:List, x:ElType) → IdxType
{ Mencari apakah ada elemen List l yang bernilai x. }
{ Jika ada, menghasilkan indeks i terkecil, di mana elemen l ke-i = x.
  Jika tidak ada, mengirimkan indeks tak terdefinisi (idxUndef).
  Jika list kosong, menghasilkan indeks tak terdefinisi (idxUndef). }
{ Memakai skema searching tanpa boolean. }
```

Latihan Soal 1

```
procedure insertUnique (input/output l:List, input x:ElType)
{ Menambahkan x sebagai elemen terakhir list l,
  pada list dengan elemen unik. }
{ I.S. List l boleh kosong, tetapi tidak penuh
  dan semua elemennya bernilai unik, tidak terurut.
  F.S. Menambahkan x sebagai elemen terakhir l
       jika belum ada elemen yang bernilai x.
       Jika sudah ada elemen list yang bernilai x
       maka I.S. = F.S. dan dituliskan pesan
       “nilai sudah ada”. }
{ Proses : Cek apakah X ada dengan sequential search
  dengan sentinel, kemudian tambahkan jika belum ada. }
```

Latihan Soal 2

Jika ADT List pada soal 1 diubah menjadi representasi **implisit**, apa yang harus diubah dari struktur data berikut?

```
{  Versi I : Dengan banyaknya elemen secara eksplisit,
             array statik }
constant kapasitas: integer = 100
constant idxUndef: integer = -1 { indeks tak terdefinisi}
{ Definisi elemen dan koleksi objek }
type ElType: integer   { type elemen list }
type List: < ti: array [0..kapasitas-1] of ElType,
                                { memori tempat penyimpan elemen (container) }
           nEff: integer ≥ 0 { banyaknya elemen efektif } >
```

Latihan Soal 3

Buatlah fungsi `getFirstIdx` dan `getLastIdx` dalam representasi array secara **eksplisit** dan **implisit**.

Kerjakan kembali fungsi-fungsi yang dikerjakan di latihan soal 1, tetapi untuk representasi implisit. Bagian kode mana saja yang berubah?

Latihan Soal 4a: ADT Array Eksplisit-Statik

```
procedure closestPair (input l: List; output p1,p2: ElType)
{ I.S.: l terdefinisi, mungkin kosong, p1 dan p2 sembarang. }
{ F.S.:
    Jika l tidak kosong, p1 dan p2 berisi 2 elemen l pada posisi
    berurutan yang memiliki selisih (selalu positif) terkecil.
    Jika kedua elemen nilainya berbeda, maka p1 adalah elemen yang
    bernilai lebih kecil.
    Jika ada beberapa pasang elemen yang memiliki selisih terkecil,
    maka diambil pasangan elemen yang muncul pertama kali.
    Jika l kosong atau hanya terdiri atas 1 elemen, p1 dan p2
    berisi elemen tidak terdefinisi yaitu -999 }
{ Contoh:
  l.ti = [5,3,10,11,20,19];    maka p1=10 dan p2=11
  l.ti = [-2,10,7,30,40,43,9]; maka p1=7  dan p2=10
  l.ti = [-2,10,10,40,40];     maka p1=10 dan p2=10 }
```

Latihan Soal 4b: ADT Array Eksplisit-Statik

```
function isFront (l1,l2 : List) → boolean
{ Mengembalikan true jika elemen-elemen l1 merupakan bagian awal dari l2 }
{ Contoh: }
{     isFront ([2,3,4], [2,3,4,5,6]) = true }
{     isFront ([2,3,4], [3,4,5,6]) = false }
{     isFront ([], [2,3,4,5,6]) = true }
{     isFront ([2,3,4], [2,3]) = false }
{     isFront ([2,3,4], []) = false }
```