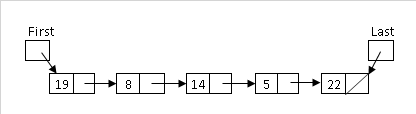
**BAGIAN 1**

Terdefinisi ADT sebuah **list berkait tunggal** sebagai berikut:



ADT

type adrSingle: pointer to elmtSingle

type elmtSingle: <nilai: integer,

next: adrSingle>

type listSingle: <First, Last: adrSingle>

1. Buat prosedur di bawah ini:

procedure insertLast (input/output L:listSingle, input newNilai: integer)

{I.S. L mungkin kosong

F.S. newNilai ada pada L sebagai elemen terakhir}

1. Buat fungsi di bawah ini

function searchNilai(L:listSingle, x:integer) 🡪 boolean

{Mencari nilai X padal list L. Fungsi mengirimkan true jika x ada pada L atau false jika tidak ada}

1. Buat prosedur di bawah ini

procedure createListSingle(output L:listSingle)

{I.S. –

Process: menerima bilangan bulat secara berulang. Untuk setiap bilangan positif yang diterima akan dimasukkan ke dalam list sebagai elemen terakhir **jika bilangan tersebut belum ada pada list.** Proses berhenti jika dimasukkan bilangan 0 atau negative.

F.S. L mempunyai sejumlah elemen yang menyimpan bilangan-bilangan positif yang unik atau kosong}

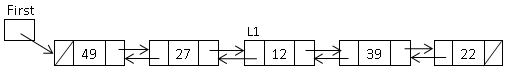
1. Buat fungsi berikut

function banyakElemen(L:listSingle) 🡪 integer

{Fungsi mengirimkan banyaknya elemen pada list L}

**BAGIAN 2**

Terdefinisi ADT sebuah **list berkait ganda** sebagai berikut:



ADT

type adrDouble: pointer to elmtDouble

type elmtDouble: <nilai: integer,

prev, next: adrDouble>

type listDouble: <First: adrDouble>

1. Buat prosedur di bawah ini:

procedure insertFirst (input/output L:listDouble, input newNilai: integer)

{I.S. L mungkin kosong

F.S. newNilai ada pada L sebagai elemen pertama}

1. Buat prosedur di bawah ini

procedure createListDouble(input N: integer, output L:listDouble)

{I.S. –

Process: menerima bilangan bulat dari piranti masukan (user) sebanyak N. Untuk setiap bilangan yang diterima akan dimasukkan ke dalam list sebagai elemen pertama**.**

F.S. L mempunyai N elemen}

**BAGIAN 3**

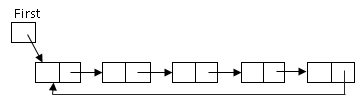
Pada program utama:

1. A. Panggil prosedur createListSingle.

B. Panggil prosedur createListDouble dengan **jumlah elemen sebanyak jumlah elemen pada list single pada point A**.

**BAGIAN 4**

Terdefinisi ADT sebuah **list berkait sirkuler** sebagai berikut:



ADT

type adrSirkuler: pointer to elmtSirkuler

type elmtSirkuler: <nilai: integer,

next: adrSirkuler>

type listSirkuler: <First: adrSirkuler>

1. Buat prosedur di bawah ini:

procedure insertLast (input/output L:listSirkuler, input newNilai: integer)

{I.S. L mungkin kosong

F.S. newNilai ada pada L sebagai elemen terakhir}

1. Buat prosedur di bawah ini

procedure createListSirkuler(input L1:listSingle, L2:listDouble;

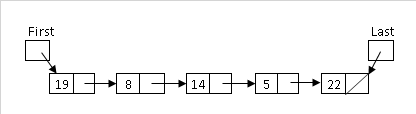
output L3: listSirkuler)

{I.S. Terdefinisi L1 dan L2. L1 dan L2 mempunyai jumlah elemen yang sama.

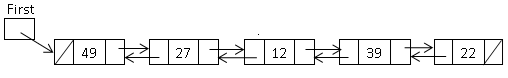
F.S. Terdefinisi L3 yang setiap elemennya merupakan jumlah elemen pada L1 dan L2}

Contoh:

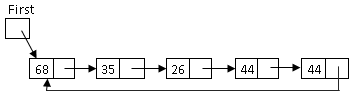
L1:



L2:



Maka L3:



1. Panggil prosedur createListSirkuler pada program utama. Tampilkan semua elemen dari setiap list yang telah dibuat ?(L1, L2, L3).

**Lampiran: algoritma selection (MAKSIMUM) sort untuk array**

|  |
| --- |
| **procedure** maxSort (input/output t: intArray)  {I.S. TabInt terdefinisi dengan N element, N>0.}  {F.S. TabInt terurut **mengecil/descending**} |
| **kamus**  i, pass, temp, iMax: integer |
| **algoritma**  pass 🡨 1  while pass < T.N-1 do  iMax 🡨 pass  i 🡨 pass +1  while i < T.N do  ifT.tabInt[iMax] < T.tabInt[i] then  iMax 🡨 i  i 🡨 i +1  *{endwhile, i > T.N}*  *{Tukar}*  temp 🡨 T.tabInt[iMax]  T.tabInt[iMax] 🡨 T.tabInt[pass]  T.tabInt[pass] 🡨 temp  pass 🡨 pass +1  *{endwhile, pass > T.N-1}* |

**LinK pengisian feedback asprak :**

**bit.ly/FeedbackPelayananIFLAB2021**

**bit.ly/SurveyKenyamananPraktikan2021**