Praktikum StrukturData
Modul PSDA

PSDA-07. VARIASI LIST LINIER

ADT List First-Last dengan Dummy pada Last

1. Buatlah ADT List yang memiliki penunjuk First dan Last, dengan elemen *dummy* pada *last element*, dengan dengan representasi fisik pointer, dalam bahasa C sesuai spesifikasi di bawah ini. Untuk type boolean, gunakan file berisi definisi type boolean secara terpisah (**boolean.h**).
   Elemen dummy di-create di awal, dan elemen ini selalu menjadi elemen dummy.
2. Buatlah driver untuk memeriksa apakah semua primitif yang didefinisikan telah berjalan dengan baik.

```
{ ADT list dummy last, berkait dengan representasi fisik pointer }
{ Dengan 2 penunjuk, first dan last, dummy selalu menunjuk last }
{ Representasi address dengan pointer }
{ infotype adalah integer }
{ Dummy dialokasi pada saat create list, dan selalu menjadi elemen dummy }

{ *** Konstanta *** }
constant Nil : ...

{ *** Definisi Type List *** }
type infotype : integer
type address  : pointer to ElmtList
type ElmtList : < info : infotype,
                  next : address >

{ Definisi list : }
{ List kosong : First(L) = dummy@ dan Last(L) = dummy@ }
{ Setiap elemen dengan address P dapat diacu Info(P), Next(P) }
{ Elemen dummy terletak pada last }

type List : < First : address,
              Last : address   >

{ Definisikan selektor yang tepat }

{ ***************** PRIMITIF-PRIMITIF LIST ***************** }

{ *** TEST LIST KOSONG *** }
function ListEmpty (L : List) → boolean
{ Mengirim true jika list kosong: First(L) = dummy@ dan Last(L) = dummy@ }

{ *** PEMBUATAN LIST KOSONG *** }
procedure CreateList (output L : List)
{ I.S. Sembarang }
{ F.S. Terbentuk list L kosong, dengan satu elemen dummy }

{ *** Manajemen Memori *** }
function Alokasi (X : infotype) → address
{ Mengirimkan address hasil alokasi sebuah elemen }
{ Jika alokasi berhasil, maka address tidak Nil, dan misalnya menghasilkan P, maka
   Info(P) = X, Next(P) = Nil }
```

```
{ Jika alokasi gagal, mengirimkan Nil }
procedure Dealokasi (input/output P : address)
{ I.S. P terdefinisi }
{ F.S. P dikembalikan ke sistem }
{ Melakukan dealokasi/pengembalian address P }

{ *** PENCARIAN SEBUAH ELEMEN LIST *** }
function FSearch (L : List, P : address) → boolean
{ Mencari apakah ada elemen list yang beralamat P }
{ Mengirimkan true jika ada, false jika tidak ada }
function Search (L : List, X : infotype) → address
{ Mencari apakah ada elemen list dengan Info(P) = X }
{ Jika ada, mengirimkan address elemen tersebut }
{ Jika tidak ada, mengirimkan Nil }

{ ***************** PRIMITIF BERDASARKAN NILAI *****************}
{ *** PENAMBAHAN ELEMEN *** }
procedure InsVFirst (input/output L : List, input X : infotype)
{ I.S. L mungkin kosong }
{ F.S. Melakukan alokasi sebuah elemen dan menambahkan elemen pertama dengan nilai
       X jika alokasi berhasil }
procedure InsVLast (input/output L : List, input X : infotype)
{ I.S. L mungkin kosong }
{ F.S. Melakukan alokasi sebuah elemen dan menambahkan elemen list di sebelum
       elemen akhir (elemen sebelum elemen dummy) bernilai X
        jika alokasi berhasil. }
{      Jika alokasi gagal: I.S. = F.S. }

{ *** PENGHAPUSAN ELEMEN *** }
procedure DelVFirst (input/output L : List, output X : infotype)
{ I.S. List L tidak kosong  }
{ F.S. Elemen pertama list dihapus : nilai info disimpan pada X }
{      dan alamat elemen pertama didealokasi }
procedure DelVLast (input/output L : List, output X : infotype)
{ I.S. List tidak kosong }
{ F.S. Elemen sebelum dummy dihapus : nilai info disimpan pada X }
{      dan alamat elemen terakhir sebelum dummy di-dealokasi }

{ ***************** PRIMITIF BERDASARKAN ALAMAT ***************** }
{ *** PENAMBAHAN ELEMEN BERDASARKAN ALAMAT *** }
procedure InsertFirst (input/output L : List, input P : address)
{ I.S. Sembarang, P sudah dialokasi  }
{ F.S. Menambahkan elemen ber-address P sebagai elemen pertama }
procedure InsertAfter (input/output L : List, input P : address, input Prec :
address)
{ I.S. Prec pastilah elemen list dan bukan elemen terakhir, }
{      P sudah dialokasi  }
{ F.S. Insert P sebagai elemen sesudah elemen beralamat Prec }
procedure InsertLast (input/output L : List, input P : address)
{ I.S. Sembarang, P sudah dialokasi  }
{ F.S. P ditambahkan sebagai elemen terakhir yang baru, }
{      yaitu menjadi elemen sebelum dummy }

{ *** PENGHAPUSAN SEBUAH ELEMEN *** }
procedure DelFirst (input/output L : List, output P : address)
{ I.S. List tidak kosong }
{ F.S. P adalah alamat elemen pertama list sebelum penghapusan }
```

```
{       Elemen list berkurang satu (mungkin menjadi kosong) }
{ First element yg baru adalah suksesor elemen pertama yang lama }
procedure DelLast (input/output L : List, output P : address)
{ I.S. List tidak kosong }
{ F.S. P adalah alamat elemen terakhir sebelum dummy pada list sebelum
       penghapusan  }
{       Elemen list berkurang satu (mungkin menjadi kosong) }
procedure DelAfter (input/output L : List, output Pdel : address, input Prec :
address)
{ I.S. List tidak kosong. Prec adalah anggota list. }
{ F.S. Menghapus Next(Prec) : Pdel adalah alamat elemen list yang dihapus  }

{ ***************** PROSES SEMUA ELEMEN LIST ***************** }
procedure PrintInfo (input L : List)
{ I.S. List mungkin kosong }
{ F.S. Jika list tidak kosong, }
{ Semua info yg disimpan pada elemen list (kecuali dummy) diprint }
{ Jika list kosong, hanya menuliskan "list kosong" }
```