F-10. ADT Pohon Biner

Buatlah ADT pohon biner sesuai dengan definisi dan spesifikasi di bawah ini dalam Bahasa LISP. Sebagai referensi dapat dilihat Diktat Dasar Pemrograman Bagian Pemrograman Fungsional dan Diktat LISP.

TYPE POHON BINER
DEFINISI DAN SPESIFIKASI TYPE
type Elemen : integer
type PohonBiner : < A : Elemen, L : PohonBiner, R : PohonBiner > {notasi prefix }
{ Pohon Biner terdiri dari Akar yang berupa elemen, L dan R adalah Pohon biner yang merupakan subPOhon kiri dan subpohon kanan }
DEFINISI DAN SPESIFIKASI SELEKTOR
Akar : PohonBiner tidak kosong → Elemen
{ Akar(P) adalah Akar dari P. Jika P adalah //L A R\\ = Akar(P) adalah A }
Left : PohonBiner tidak kosong → PohonBiner
{ Left(P) adalah sub pohon kiri dari P. Jika P adalah //L A R\ \, Left (P) adalah L }
Right : PohonBiner tidak kosong → PohonBiner
{ Right(P) adalah sub pohon kanan dari P. Jika P adalah //L A R\\,Right (P) adalah R}
DEFINISI DAN SPESIFIKASI KONSTRUKTOR
{ Perhatikanlah bahwa konstruktor pohon biner dengan basis pohon kosong dituliskan sebagai
a. Infix : //L A R
b. Prefix : //A L R
c. Posfix : //L R A
atau bahkan notasi lain yang dipilih }
DEFINISI DAN SPESIFIKASI PREDIKAT
IsTreeEmpty : PohonBiner → boolean
{ IsTreeEmpty(P) true jika P kosong : (// \\) }
IsOneElmt : PohonBiner → boolean
{ IsOneElement(P) true jika P hanya mempunyai satu elemen, yaitu akar (// A \\\) }
IsUnerLeft : PohonBiner → boolean
{ IsUnerLeft(P) true jika P hanya mengandung sub pohon kiri tidak kosong: (//L A \\) }
IsUnerRight : PohonBiner → boolean
{ IsUnerRight (P) true jika P hanya mengandung sub pohon kanan tidak kosong: (//A R\\) }
IsBiner : PohonBiner tidak kosong → boolean
{ IsBiner(P) true jika P mengandung sub pohon kiri dan sub pohon kanan : (//L A R\\) }
IsExistLeft : PohonBiner tidak kosong → boolean
{ IsExistLeft (P) true jika P mengandung sub pohon kiri }
IsExistRight : PohonBiner tidak kosong → boolean
{ IsExistRight(P) true jika P mengandung sub pohon kanan }
DEFINISI DAN SPESIFIKASI OPERASI LAIN
NbElmt : PohonBiner → integer ≥ 0
{ NbElmt(P) memberikan banyaknya elemen dari pohon P.P boleh kosong. }
NbDaun : PohonBiner → integer ≥ 0
{ NbDaun (P) memberikan banyaknya daun dari pohon P. P boleh kosong }
NbDaun1 : PohonBiner → integer ≥ 1

{ Prekondisi : Pohon P tidak kosong }
{ NbDaun1 (P) memberikan Banyaknya daun dari pohon P }
RepPrefix: PohonBiner → list of elemen
{ RepPrefix (P) memberikan representasi linier (dalam bentuk list), dengan urutan elemen list sesuai dengan urutan penulisan pohon secara prefix. P boleh kosong. }

IsMember : PohonBiner, elemen → boolean
{IsMember(P,X) mengirimkan true jika ada node dari P yg bernilai X }
IsSkewLeft: PohonBiner → boolean
{ IsSkewLeft(P) mengirimkan true jika P adalah pohon condong kiri }
IsSkewRight : PohonBiner → boolean
{ IsSkewRight(P) mengirimkan true jika P adalah pohon condong kiri }
LevelOfX: PohonBiner, elemen → integer
{ LevelOfX(P,X) Mengirimkan level dari node X yang merupakan salah satu simpul dari pohon biner P }
AddDaunTerkiri : PohonBiner, elemen → PohonBiner
{ AddDaunTerkiri(P,X): mengirimkan Pohon Biner P yang telah bertambah simpulnya, dengan X sebagai simpul daun terkiri }
AddDaun : PohonBiner tidak kosong, elemen, elemen, boolean → PohonBiner
{ AddDaun (P,X,Y,Kiri) : P bertambah simpulnya, dengan Y sebagai anak kiri X (jika Kiri), atau sebagai anak Kanan X (jika not Kiri)
{ Prekondisi : X adalah salah satu daun Pohon Biner P }
DelDaunTerkiri: PohonBiner tidak kosong → <PohonBiner, elemen>
{ DelDaunTerkiri(P) menghasilkan sebuah pohon yang dihapus daun terkirinya, dengan X adalah info yang semula disimpan pada daun terkiri yang dihapus }
DelDaun : PohonBiner tidak kosong, elemen → PohonBiner
{ DelDaun(P,X) dengan X adalah salah satu daun , menghasilkan sebuah pohon tanpa X yang semula adalah daun dari P}
MakeListDaun : PohonBiner → list of elemen
{ MakeListDaun(P) : Jika P adalah pohon kosong, maka menghasilkan list kosong. Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua daun pohon P}
MakeListPreOrder : PohonBiner → list of elemen
{ MakeListPreOrder(P) : Jika P adalah pohon kosong, maka menghasilkan list kosong. }
{ Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua node pohon P dengan urutan Preorder }
MakeListPostOrder : PohonBiner → list of elemen
{ MakeListPostOrder(P) : Jika P adalah pohon kosong, maka menghasilkan list kosong. }
{ Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua node pohon P dengan urutan PostOrder }
MakeListInOrder : PohonBiner → list of elemen
{ MakeListInOrder(P) : }
{ Jika P adalah pohon kosong, maka menghasilkan list kosong. }
{ Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua node pohon P dengan urutan InOrder }
MakeListLevel : PohonBiner, integer → list of elemen
{ MakeListLevel(P,N) : }

{ Jika P adalah pohon kosong, maka menghasilkan list kosong. }
{ Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua node pohon P yang levelnya=N }