IMPLEMENTASI LIST

Menambah Elemen List

```
def konso(S,L):
    if L==[]:
        return [S]
    else:
        return [S]+L

def konsi(S,L):
    if L==[]:
        return [S]
    else:
        return L+[S]
```

DEFINISI DAN SPESIFIKASI KONSTRUKTOR

Konso : elemen, List $\rightarrow$ List

{Konso(e,L): menghasilkan sebuah list dari e dan L,
: $e \; o \; L \rightarrow L'$}

Kons• : List, elemen $\rightarrow$ List

{ Kons(L,e): menghasilkan sebuah list dari L dan list : $L \bullet e \rightarrow L'$*}*

Cek elemen List

{Basis 1 }

IsOneElmt: List → boolean

{IsOneElmt (X,L) adalah benar jika list L hanya mempunyai satu elemen }

```
def is_one_element(L):
    if not(is_empty(L)):
        return NB_element(L)==1
```

Keanggotaan List-1

KEANGGOTAAN **IsMember(x,L)**

DEFINISI DAN SPESIFIKASI

IsMember (x,L) : elemen, List → boolean

{ IsMember (x,L) true jika x adalah elemen dari list L }

REALISASI VERSI-2 : DENGAN KONSO

{ Basis 0 : List kosong: tidak mengandung apapun, → false

Rekurens:

e	*o*	*Tail(L)*

? *x*

Kasus: *e=x → true*

e ≠ x → x adalah anggota Tail(L) }

```
IsMember(x,L) :
          if IsEmpty(x) then {Basis 0}
                false
          else {Rekurens : analisa kasus}
              if FirstElmt(L)=x then true
                        else IsMember(x,Tail (L))
```

```
def is_member(L,x):
    if is_empty(L):
        return False
    else:
        if first_element(L)==x:
            return True
        else:
            return is_member(tail(L),x)
```

Keanggotaan List-2

REALISASI VERSI-2 : DENGAN KONS•

{ Basis 0: list kosong tidak mengandung apapun, → false

Rekurens: Head(L) • e

x

e=x → true

e≠x → elemen Head(L) }

```
IsMember (x,L) :
  if IsEmpty(x) then {Basis 0}
                     false
  else {Rekurens: analisa kasus }
     if LastElmt(L)=x then true
      else IsMember (x,Head (L))
```

```
def is_member (L,x):
    if is_empty(L):
        return False
    else:
        if last_element(L)==x:
            return True
        else:
            return is_member (head(L),x)
```

Menyalin List

MENYALIN LIST **Copy(L)**

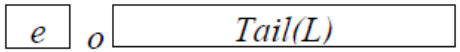
DEFINISI DAN SPESIFIKASI

Copy : List → List

{Copy (L) menghasilkan salinan list L , artinya list lain yang identik dengan L}

REALISASI: DENGAN KONSO

{ Basis 0 : list kosong: hasilnya list kosong
Rekurens:



e o Copy(Tail(L)) }

```
Copy (L) :
        if IsEmpty(L) then {Basis 0}
                []
        else {Rekurens}
              Konso(FirstElmt(L), Copy(Tail (L))
```

```
def copy_List(L):
    if is_empty(L):
        return []
    else:
        return konso(first_element(L),copy_List(tail(L)))
```

Membalik urutan List

Inverse ([ ]) = [ ] ; Inverse ([a, b, c]) = [c, b, a]

MEMBALIK LIST **Inverse(L)**

DEFINISI DAN SPESIFIKASI

Inverse (L) : List → List

{Inverse (L) menghasilkan salinan list L dengan urutan elemen terbalik}

REALISASI: DENGAN KONSO

{ Basis 0: list kosong: hasilnya list kosong

Rekurens:

e	o	Tail(L)

Hasil pembalikan adalah Tail(L) • e }

```
Inverse(L) :
        if IsEmpty(L) then {Basis 0}
            []
        else {Rekurens}
            Kons•( Inverse(Tail (L), FirstElmt(L))
```

```
if is_empty(L):
    return []
else:
    return konsi(first_element(L),invers_List(tail(L)))
```

Concatenate List

KONKATENASI **Concat(L1,L2)**

DEFINISI DAN SPESIFIKASI

Concat (L1,L2) : List → List

{Concat (L1,L2)menghasilkan konkatenasi list L1 dan L2}

REALISASI : REKURENS TERHADAP L1

{ Basis 0: L1 [] : L2

Rekurens:

e1 o *Tail(L1)*

L2

List Hasil: e1 o Hasil konkatenasi dari Tail(L1) dengan L2

```
Konkat(L1,L2) :
if IsEmpty(L1) then {Basis 0}
      L2
else {Rekurens}
       Konso(FirstElmt(L1), Konkat(Tail(L1),L2) )
```

```
def concatenate_List(L1,L2):
    if is_empty(L1):
        return L2
    else:
        return konso(first_element(L1),concatenate_List(tail(L1),L2))
```

Tugas 1-1

ELEMEN KE N — **ElmtKeN(N,L)**

DEFINISI DAN SPESIFIKASI

ElmtKeN (N,L) : integer ≥ ,List tidak kosong→ elemen

{ElmtKeN (L) menghasilkan elemen ke-N list L, N ≥ 0, dan N ≤ banyaknya elemen list. }

REALISASI: DENGAN KONSO

{ Basis 1 : List dengan satu elemen , dan N=1 : elemen pertama list

e

Rekurens:

e	*o*	*Tail(L)*

1 + ----------N-1------

---------- N ----------

Kasus : N=1 maka e

N>1 : bukan e, tetapi ElmtKeN (N-1, Tail(L))

}

```
ElmtKeN(N,L) :
        if N=1 {Basis 1}then
               FirstElmt(L)
        else {Rekurens}
             ElmtKeN(prec(N),Tail (L))
```

Tugas 1-2

APAKAH X ELEMEN KE N **IsXElmtKeN(X,N,L)**

DEFINISI DAN SPESIFIKASI

IsXElmtKeN (N,L) :elemen, integer ≥ 0, List (tidak kosong) → boolean

{IsXElmtKeN (L) true jika X adalah elemen ke-N list L, N ≥ 0, dan N ≤ banyaknya elemen list false jika tidak}

REALISASI: DENGAN KONSO

{ Basis 0: List dengan satu elemen , dan N=1dan e=X : true

e

Rekurens:

e	*o*	*Tail(L)*

e =X ----------N-1-------

----------N ----------

IsXElmtKeN (X,N-1, Tail(L))

```
IsXElmtKeN(X,N,L) :
        if IsMember (X,L) then {Analisa kasus }
             if N=1 and FirstElmt(L)=X then {Basis 0}
                   true
              else {Rekurens}
                   false or IsXElmtKeN(X,prec(N),Tail (L))
        else {Bukan member, pasti } false
```

REALISASI:

```
{ Realisasi ini memanfaatkan fungsi ElmtKeN(L) yang sudah didefinisikan
}
IsXElmtKeN(X,L,N) :
            ElmtKeN(N,L)=X
```

Tugas 1-3

APAKAH INVERSE **IsInverse(L1,L2)**

DEFINISI DAN SPESIFIKASI

IsInverse (L1,L2) : 2 List → boolean

{IsInverse (L1,L2) true jika L2 adalah list dengan urutan elemn terbalik dibandingkan L1, dengan perkataan lain adalah hasil inverse dari L1}

REALISASI: DENGAN NAMA DAN FUNGSI ANTARA

```
IsInverse(L) :

            IsEqual (L3,L2)
```

REALISASI LAIN

{ Basis 1 : list dengan satu elemen : true
Rekurens: dua buah list sama, jika panjangnya sama dan

L1 : [*e1*] *o* [*Tail(L1) -x1*] • [*x1*]

L2 : [*e2*] *o* [*Tail(L2) – x2*] • [*x2*]

e1=x2 dan Tail(L1) -x1 = Tail(L2)-x2

```
IsInverse(L) :
       if NbElmt(L1) = NbELmt(L2) then {Analisa kasus }
         if IsEmpty(L1) and IsEmpty(L2) then {Basis 0}
                true
         else {Rekurens }
                ( FirstElmt(L1)=LastElmt(L2)) and
                 IsInverse (Head(Tail(L1)), Head(Tail(L2))
      else{panjang tidak sama, pasti bukan hasil inverse }
          false
```