LIST OF LIST

DASAR PEMROGRAMAN

PRAKTIKUM KE-10

DEFINISI LIST OF LIST

List of list adalah list yang :

- mungkin kosong,
- mungkin terdiri dari sebuah elemen yang disebut **atom dan sisanya adalah list of list**,
- mungkin terdiri dari sebuah elemen berupa **list dan sisanya adalah list of list**.

Untuk membedakan antara list dengan atom:
List dituliskan di antara tanda kurung [ ], sedangkan Atom dituliskan tanpa tanda kurung.

TYPE LIST OF LIST

DEFINISI DAN SPESIFIKASI PREDIKAT KHUSUS UNTUK LIST OF LIST

IsEmpty : list of list → boolean

{IsEmpty(S) benar jika S adalah list of list kosong}

IsAtom : list of list → boolean

{IsAtom(S) menghasilkan true jika list adalah atom, yaitu terdiri dari sebuah atom }

IsList : list of list → boolean

{ IsList(S) menghasilkan true jika S adalah sebuah list (bukan atom)}

TYPE LIST OF LIST

```
def IsEmptyLoL(S):
    return S==[]

def IsAtom(S):
    return type(S) != list

def IsList(S):
    return not IsAtom(S)
```

TYPE LIST OF LIST

DEFINISI DAN SPESIFIKASI KONSTRUKTOR

KonsLo : List, List of list → List of list

{ KonsLo(L,S) diberikan sebuah List L dan sebuah List of List S, membentuk list baru dengan List yang diberikan sebagai elemen pertama List of list: L o S → S'}

KonsL• : List of list , List → List of list

{KonsL• (S,L) diberikan sebuah List of list S dan sebuah list L, membentuk list baru dengan List yang diberikan sebagai elemen terakhir list of List: S • L → S'}

TYPE LIST OF LIST

```
1  def KonsoLoL(L,S):
2      if IsEmptyLoL(S):
3          return [L]
4      else:
5          return [L] + S
```

```
1  def KonsiLoL(L,S):
2      if IsEmptyLoL(S):
3          return [L]
4      else:
5          return S + [L]
```

TYPE LIST OF LIST

DEFINISI DAN SPESIFIKASI SELEKTOR

FirstList: List of list tidak kosong → List

{FirstList(S) Menghasilkan elemen pertama list, mungkin sebuah list atau atom }

TailList : List of list tidak kosong → List of list

{TailList(S) Menghasilkan "sisa" list of list S tanpa elemen pertama list S }

LastList : List of list tidak kosong → List of list

{LastList(S) : Menghasilkan elemen terakhir list of list S, mungkin list atau atom }

HeadList : List of list tidak kosong → List of list

{HeadList(S) Menghasilkan "sisa" list of list tanpa elemen terakhir list }

TYPE LIST OF LIST

```
1  def FirstList(S):
2      if not(IsEmptyLoL(S)):
3          return S[0]
4
5  def TailList(S):
6      if not(IsEmptyLoL(S)):
7          return S[1:]
```

```
1  def LastList(S):
2      if not(IsEmptyLoL(S)):
3          return S[-1]
4
5  def HeadList(S):
6      if not(IsEmptyLoL(S)):
7          return S[:-1]
```

KESAMAAN 2 BUAH LIST OF LIST

KESAMAAN **IsEqS (L1,S2)**

DEFINISI PREDIKAT

IsEqS : 2 List of list → boolean

{IsEqS (S1,S2) true jika S1 identik dengan S2 : semua elemennya sama }

{ Basis : kedua list kosong : → true

salah satu list kosong : → false

Rekurens :

S1 *L1* ○ *Tail(S1)*

S2 *L2* ○ *Tail(S2)*

L1 dan L2 adalah atom : L1=L2 and IsEqS(TailList(S1),TailList(S2))

L1 dan L2 adalah list : IsEqS(S1,S2) and IsEqS(TailList(S1),TailList(S2))

else : false

}

KESAMAAN 2 BUAH LIST OF LIST

REALISASI

```
IsEqS(S1,S2) :
    depend on S1, S2
        IsEmpty(S1) and IsEmpty(S2) : true
        not IsEmpty(S1) and IsEmpty(S2) : false
        IsEmpty(S1) and not IsEmpty(S2) : false
        not IsEmpty(S1) and not IsEmpty(S2) :
          depend on FirstList(S1), FirstList(S2)
              IsAtom(FirstList(S1) and IsAtom(FirstList(S1)) :
                    FirstList(S1) = FirstList(S1) and
                    IsEqS (TailList(S1), TailList(S2))
              IsList(FirstList(S1) and IsList(FirstList(S1):
                   IsEqS(FirstList(S1),FirstList(S2)) and
                   IsEqS(TailList(S1),TailList(S2))
                Else {atom dengan list pasti tidak sama}
                    false
```

KESAMAAN 2 BUAH LIST OF LIST

```
IsEqS([], [[4], 5])
False

IsEqS([1,[2]], [1,[2]])
1 == 1 and IsEqS([[2]], [[2]])
True and IsEqS([2], [2]) and IsEqS([], [])
True and 2 == 2 and IsEqS([], []) and True
True and True and True and True
True

IsEqS([3,[4]], [[4], 3])
False
```

KEANGGOTAAN ATOM DALAM LIST OF LIST

KEANGGOTAAN **IsMemberS (A,S)**

DEFINISI PREDIKAT

IsMemberS : elemen, List of list → boolean

{ IsMemberS (A,S) true jika A adalah anggota S }
{ Basis : list kosong : → false
Rekurens :

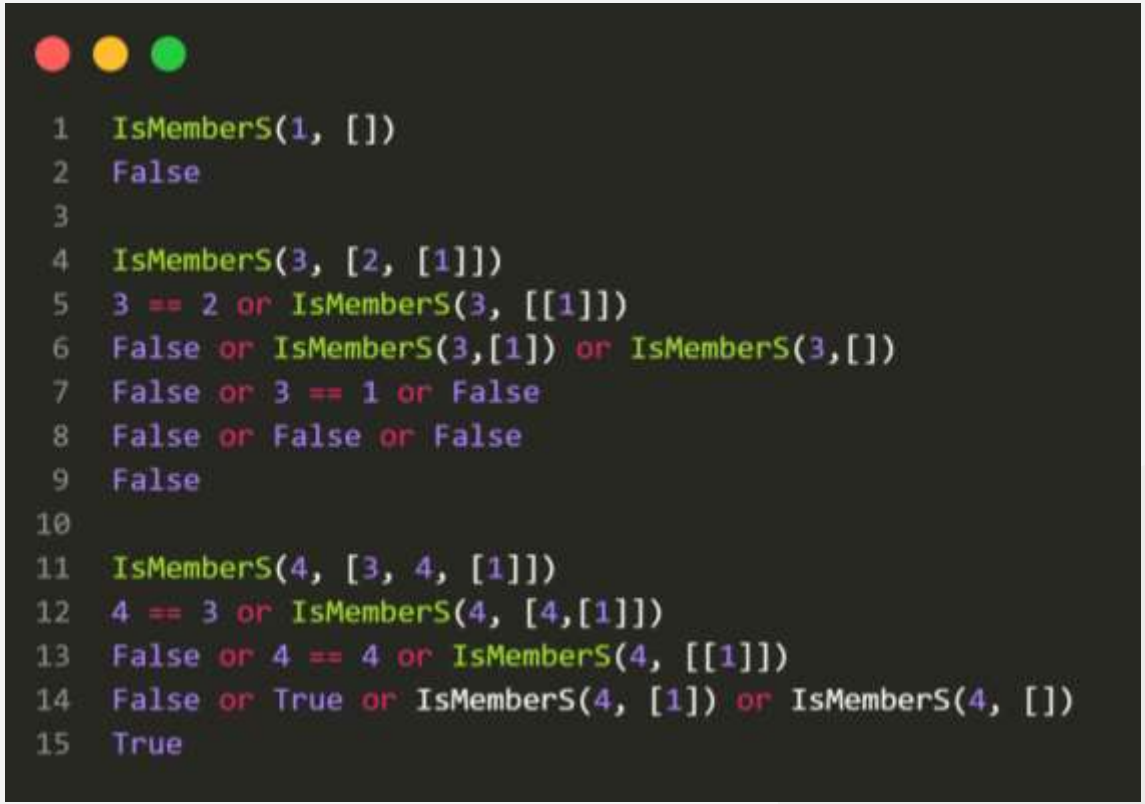
L1	○	Tail(S)

L1 adalah atom dan A = L1 : true
L1 bukan atom : A anggota L1 or IsMemberS(A, TailList(S))
}

REALISASI

```
IsMemberS(A,S) :
   depend on S
        IsEmpty(S): false
        Not IsEmpty(S) :
            depend on FirstList(S)
                IsAtom(FirstList(S)): A = FirstList(S)
                IsList(FirstList(S)) : IsMember(A,FirstList(S))
                   or IsMemberS(A,TailList(S))
{ dengan IsMember(A,L) adalah fungsi yang mengirimkan true jika A adalah
elemen list L}
```

KEANGGOTAAN ATOM DALAM LIST OF LIST



```
1  IsMemberS(1, [])
2  False
3
4  IsMemberS(3, [2, [1]])
5  3 == 2 or IsMemberS(3, [[1]])
6  False or IsMemberS(3,[1]) or IsMemberS(3,[])
7  False or 3 == 1 or False
8  False or False or False
9  False
10
11 IsMemberS(4, [3, 4, [1]])
12 4 == 3 or IsMemberS(4, [4,[1]])
13 False or 4 == 4 or IsMemberS(4, [[1]])
14 False or True or IsMemberS(4, [1]) or IsMemberS(4, [])
15 True
```

KEANGGOTAAN LIST DALAM LIST OF LIST

KEANGGOTAAN **IsMemberLS (L,S)**

DEFINISI PREDIKAT

IsMemberLS : List, List of list → boolean

{ IsMemberLS (L,S2) true jika L adalah anggota S }
{ Basis : L dan S list kosong : → true
L atau S tidak kosong : false
Rekurens :

L1 ○ Tail(S)

L1 adalah atom : IsMemberLS(L,TailList(S))
L1 bukan atom : L1=L : true
L1≠L : IsMemberLS(L,TailList(S))
}

REALISASI

```
IsMemberLS(L,S) :
   depend on S
        IsEmpty(L) and IsEmpty(S): true
        not IsEmpty(L) and IsEmpty(S) : false
        IsEmpty(L) and not IsEmpty(S) : false
         not IsEmpty(L) and not IsEmpty(S) :
            if (IsATOM(FirstList(S))) then IsMemberLS(Taillist(L,S))
             else (   IsLIST(FirstList(S)) )
                    If IsEqual(L,FirstList(S)) then true
                    else       IsMemberLS(L,Taillist(S))
{ dengan IsEqual(L1,L2) adalah fungsi yang mengirimkan true jika list L1
sama dengan list L}
```

KEANGGOTAAN LIST DALAM LIST OF LIST

```
IsMemberLS([1,2], [2,[1,2]])
IsMemberLS([1,2], [[1,2]])
True

IsMemberLS([1,2], [1,[2,1],3])
IsMemberLS([1,2], [[2,1],3])
IsMemberLS([1,2], [2,1]) or IsMemberLS([1,2], [3])
IsMemberLS([1,2], [1]) or IsMemberLS([1,2], [3])
IsMemberLS([1,2], []) or IsMemberLS([1,2], [3])
False or IsMemberLS([1,2], [3])
False or IsMemberLS([1,2], [])
False or False
False
```

MENGHAPUS SEBUAH ELEMEN LIST OF LIST

HAPUS*ELEMEN **Rember*(a,S)**

DEFINISI

Rember*: elemen, List of list → List of list

{ Rember (a,S) menghapus sebuah elemen bernilai a dari semua list S }*

{ List kosong tetap menjadi list kosong }
{ Basis : list kosong : → ( )
Rekurens :

a

S | *L1* *o* | *Tail(S)*

L1 adalah atom : L1 = a : TailList(S) tanpa a
L1 ≠ a : L1 o (TailList(S) tanpa a)
L1 adalah list : (L1 tanpa a) o (TailList(S) tanpa a)
}

REALISASI

```
Rember*(a,L) :
   if IsEmpty(S)  then S
   else if IsList(FirstList(S))
        then KonsLo(Rember*(a,FirstList(S)), Rember*(a,TailList(S)))
        else { elemen pertama S adalah atom }
             if FirstElmt(S) = a then
               Rember*(a,TailList(S))
             else
                  KonsLo (FirstElmt(S),Rember*(a,Tail(S))
```

MENGHAPUS SEBUAH ELEMEN LIST OF LIST

```
Rember(1, [3,[5,1]])
KonsoLoL(3, Rember(1, [[5,1]]))
KonsoLoL(3, KonsoLoL(Rember(1, [5,1]), Rember(1, [])))
KonsoLoL(3, KonsoLoL(KonsoLoL(5, Rember(1, [1])), Rember(1, [])))
KonsoLoL(3, KonsoLoL(KonsoLoL(5, []), Rember(1, [])))
KonsoLoL(3, KonsoLoL([5], Rember(1, [])))
KonsoLoL(3, KonsoLoL([5], []))
KonsoLoL(3, [[5]])
[3,[5]]

Rember(5, [[1],3,4])
KonsoLoL(Rember(5, [1]), Rember(5, [3,4]))
KonsoLoL(KonsoLoL(1,Rember(5, [])), Rember(5, [3,4]))
KonsoLoL(KonsoLoL(1,[]), Rember(5, [3,4]))
KonsoLoL([1], Rember(5, [3,4]))
KonsoLoL([1], KonsoLoL(3,Rember(5,[4])))
KonsoLoL([1], KonsoLoL(3,KonsoLoL(4,Rember(5,[]))))
KonsoLoL([1], KonsoLoL(3,KonsoLoL(4,[])))
KonsoLoL([1], KonsoLoL(3,[4]))
KonsoLoL([1], [3,4])
[[1],3,4]
```

ELEMEN BERNILAI MAKSIMUM DARI LOL

ELEMEN BERNILAI MAKSIMUM **Max(S)**

DEFINISI

Max List of list tidak kosong → integer

{ Max (S) menghasilkan nilai elemen (atom) yang maksimum dari S }

{ Basis : list dengan satu elemen E1
E1 adalah atom : nilai E1
E1 adalah list : Max(E1)
Rekurens :
a

S L1 *o* *Tail(S)*

L1 adalah atom : Max2(L1,Max(Tail(S))
L1 adalah list : Max2 (Max(L1), Max(Tail(S))
}
{Fungsi antara }

Max2 2 integer → integer

{Max2(a,b) menghasilkan nilai maksimum a dan b }

ELEMEN BERNILAI MAKSIMUM DARI LOL

REALISASI

```
Max2(a,b) :
    If a>=b then
        a
    Else
        b

Max(S) :
   if IsOneElmt(S)  then  {Basis 1 }
        if IsAtom(FirstList(S))  then
              FirstList(S)
        Else   { List }
              Max(FirstList(S))
   Else  {Rekurens }
       if IsAtom(FirstList(S))  then {First elemen adalah atom }
           Max2(FirstList(S),Max(TailList(S))
        Else   { Firs element adalah List }
           Max2(Max(FirstList(S)), Max(TailList(S))
```

ELEMEN BERNILAI MAKSIMUM DARI LOL

```
1  Max([1, [5, [6]], [8]])
2  Max2(1, Max([[5, [6]], [8]]))
3  Max2(1, Max2(Max([5, [6]]), Max([[8]])))
4  Max2(1, Max2(Max2(5, Max([[6]])), Max([[8]])))
5  Max2(1, Max2(Max2(5, Max([6])), Max([[8]])))
6  Max2(1, Max2(Max2(5, 6), Max([[8]])))
7  Max2(1, Max2(6, Max([[8]])))
8  Max2(1, Max2(6, Max([8])))
9  Max2(1, Max2(6, 8))
10 Max2(1, 8)
11 8
```

THANKS