

DASAR PEMROGRAMAN

SET
(HIMPUNAN)

Definisi SET

- Sebuah List yang mana elemennya hanya muncul satu kali (unik).
- List kosong adalah Set (kosong)
- Contoh :
 - [durian, pisang, mangga] adalah SET
 - [putih, merah, biru, putih, kuning] bukan SET

Type SET

TYPE SET (HIMPUNAN)

DEFINISI DAN SPESIFIKASI TYPE

{Set adalah List dengan tambahan syarat bahwa tidak ada elemen yang sama }

{ Semua konstruktor, selektor dan fungsi pada List berlaku untuk Himpunan }

DEFINISI DAN SPESIFIKASI KONSTRUKTOR HIMPUNAN DARI LIST

{ Himpunan dibentuk dari list }

MakeSet (L) : list \rightarrow set

{ membuat sebuah set dari sebuah list }

{ yaitu membuang semua kemunculan yang lebih dari satu kali }

{ List kosong tetap menjadi list kosong }

DEFINISI DAN SPESIFIKASI PREDIKAT

IsSet : list \rightarrow boolean

{ IsSet(L) true jika L adalah set }

IsSubSet : 2 set \rightarrow boolean

{ IsSubSet (H1,H2) true jika H1 adalah subset dari H2: semua elemen H1 adalah juga merupakan elemen H2 }

Type SET

DEFINISI DAN SPESIFIKASI OPERASI TERHADAP HIMPUNAN

MakeIntersect : 2 set \rightarrow set

{ Intersect (H1,H2) membuat interseksi H1 dengan H2 : yaitu set baru dengan anggota elemen yang merupakan anggota H1 dan juga anggota H2 }

MakeUnion : 2 set \rightarrow set

{ Union (H1,H2) membuat union H1 dengan H2 : yaitu set baru dengan semua anggota elemen H1 dan anggota H2 }

IsMember : elemen, list \rightarrow boolean

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

Rember : elemen, list \rightarrow list

*{ Rember (x,L) menghapus sebuah elemen bernilai x dari list }
{ list yang baru berkurang SATU elemennya yaitu yang bernilai e }
{ List kosong tetap menjadi list kosong }*

MultiRember : elemen, list \rightarrow list

*{ MultiRember (x,L) menghapus semua elemen bernilai x dari list }
{ list yang baru tidak lagi mempunyai elemen yang bernilai x }
{ List kosong tetap menjadi list kosong }*

Predikat – predikat pembentuk himpunan (SET)

- Hapus1elemen \rightarrow Rember(x,L)
- Hapus semua elemen \rightarrow MultiRember(x,L)

Menghapus sebuah elemen LIST

HAPUS1ELEMEN

Rember(e.L)

DEFINISI

Rember : elemen, list \rightarrow list

{ Rember (x,L) menghapus sebuah elemen bernilai x dari list }
{ list yang baru berkurang SATU elemennya yaitu yang bernilai e }
{ List kosong tetap menjadi list kosong }
{ Base : list kosong : \rightarrow list kosong }
Rekurens :

$$\begin{array}{c} x \\ \boxed{el} \end{array} \quad o \quad \boxed{\text{Tail}(L)}$$

e = x : hasil adalah Tail(L) ,
e \neq x : el o Hasil rember(e,Tail(L)) }

REALISASI

```
Rember (x, L) :
    if IsEmpty(L) then {Basis }
        L
    Else {Rekurens : analisa kasus }
        if FirstElmt(L)=x then Tail(L)
        else Konso (FirstElmt(L), Rember (x, Tail(L) ) )
```

REALISASI

```
Rember(x,L) :  
    if IsEmpty(L) then {Basis }  
        L  
    Else {Rekurens : analisa kasus }  
        if FirstElmt(L)=x then Tail(L)  
        else Konso (FirstElmt(L),Rember(x,Tail(L))
```

```
def Rember(x,L):  
    if is_empty(L):  
        return L  
    else:  
        if first_element(L)==x:  
            return tail(L)  
        else:  
            return konso(first_element(L),Rember(x,tail(L)))
```

L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']

Rember('a',L1) =

['d', 'v', 'a', 'd', 'ada', 'a', 'ada', 'b', 'd', 'v', 'v', 'a']

HAPUS SEMUA ELEMEN

Multiremember(e,L)

DEFINISI

MultiRember : elemen, list \rightarrow list

{ MultiRember (x,L) menghapus semua elemen bernilai x dari list }

{ list yang baru tidak lagi mempunyai elemen yang bernilai x }

{ List kosong tetap menjadi list kosong }

{ Base : list kosong : \rightarrow List kosong }

Rekurens :

$$\begin{array}{c} x \\ \boxed{e} \quad o \quad \boxed{\text{Tail}(L)} \end{array}$$

e = x : hapus semua x dari Tail(L) ,

e \neq x : e1 o hasil penghapusan semua x dari Tail(L)}

REALISASI

MultiRember(x,L) :

if IsEmpty(L) then {Basis}

L

else {Rekurens : analisa kasus }

if FirstElmt(L)=x

then MultiRember(x,Tail(L))

else Konso (FirstElmt(L),MultiRember(x,Tail(L))

REALISASI

```
MultiRember(x,L) :  
  if IsEmpty(L) then {Basis}  
    L  
  else {Rekurens : analisa kasus }  
    if FirstElmt(L)=x  
    then MultiRember(x,Tail(L))  
    else Konso (FirstElmt(L),MultiRember(x,Tail(L))
```

```
def MultiRember(x,L):  
    if is_empty(L):  
        return L  
    else:  
        if first_element(L)==x:  
            return MultiRember(x,tail(L))  
        else:  
            return konso(first_element(L),MultiRember(x,tail(L)))
```

L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']

MultiRemember('a',L1) =

['d', 'v', 'd', 'ada', 'ada', 'b', 'd', 'v', 'v']

Menguji apakah sebuah List merupakan sebuah SET?

APAKAH SET	IsSet(L)
<u>DEFINISI PREDIKAT</u>	
$\text{IsSet} : \text{list} \rightarrow \text{boolean}$	
$\{ \text{Set}(L) \text{ true jika } L \text{ adalah set} \}$	
$\{ \text{Base} : \text{list kosong adalah set}$	
Rekurens :	
<div><div>e</div><div>\circ <div>$\text{Tail}(L)$</div></div></div>	
$\text{merupakan set jika Tail}(L) \text{ tidak mengandung } e\}$	
<u>REALISASI VERSI-1</u>	
$\text{IsSet}(L) :$	
$\text{if } \text{IsEmpty}(L) \text{ then } \{ \text{Basis: list kosong adalah himpunan kosong} \}$	

REALISASI VERSI-1

```
IsSet(L) :  
    if IsEmpty(L) then {Basis: list kosong adalah himpunan kosong }  
        true  
    else {Rekurens : analisa kasus }  
        if IsMember(FirstElmt(L),Tail(L)) then false  
        else IsSet(Tail(L))
```

REALISASI VERSI-2

```
IsSet(L) :  
    if IsEmpty(S) then {Basis }  
        true  
    else {Rekurens:}  
        not IsMember(FirstElmt(L),Tail(L)) or then IsSet(Tail(L))
```

REALISASI

```
IsSet(L) :  
    Isempty(S) or then not IsMember(FirstElmt(L),Tail(L))  
        or then IsSet(Tail(L))
```

REALISASI VERSI-1

```
IsSet(L) :  
    if IsEmpty(L) then {Basis: list kosong adalah himpunan kosong }  
        true  
    else {Rekurens : analisa kasus }  
        if IsMember(FirstElmt(L),Tail(L)) then false  
        else IsSet(Tail(L))
```

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

```
L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']
```

```
L2=[2,4,7,'a','x']
```

```
print(is_set(L1)) → False
```

```
print(is_set(L2)) → True
```

Membuat SET dari sebuah List

MEMBENTUK SET (versi-1)

MakeSet (L)

DEFINISI

MakeSet1 (L) : list \rightarrow set

{ membuat sebuah set dari sebuah list }

{ yaitu membuang semua kemunculan yang lebih dari satu kali}

{ List kosong tetap menjadi list kosong }

{ Base : list kosong : \rightarrow List kosong

Rekurens :



Untuk setiap e :

e adalah Member dari Tail(L) : MakeSet(Tail(L))

e bukan Member dari Tail(L) : e o MakeSet(Tail(L)) }

REALISASI

```
MakeSet1(L) :  
  if IsEmpty(L) then {Basis}  
    L  
  else {Rekurens}  
    if IsMember (FirstElmt(L), Tail(L)) then  
      MakeSet (TAIL(L))  
    else Konso (FirstElmt(L), MakeSet (Tail(L))
```

```
def make_set(L):  
    if is_empty(L):  
        return L  
    else:  
        if is_member(tail(L), first_element(L)):  
            return make_set(tail(L))  
        else:  
            return konso(first_element(L), make_set(tail(L)))
```

```
L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']
```

```
L2=[2,4,7,'a','x']
```

```
print(is_set(L1)) → False
```

```
print(is_set(L2)) → True
```

```
L3=make_set(L1) → L3=['ada', 'b', 'd', 'v', 'a']
```

```
print(is_set(L3)) → True
```

Menguji apakah H1 SubSET H2?

APAKAH SUBSET

IsSubSet(H1,H2)

DEFINISI PREDIKAT

IsSubSet : 2 set \rightarrow boolean

IsSubSet (H1,H2) true jika H1 adalah subset dari H2: semua elemen H1 adalah juga merupakan elemen H2 }

{ List kosong adalah subset dari set apapun }

{ Base : list kosong : \rightarrow true

Rekurens :

H1 *e* o *Tail(H1)*

H2

Setiap karakter H1 harus dicek thd H2 :

e anggota dari H2 : adalah subset jika Tail(H1) adalah subset H2

e bukan anggota H2: H1 pasti bukan subset H2 }

REALISASI

```
IsSUBSet(H1,H2):  
  {Basis}  if Iseempty(H1)  then true  
  {Rekurens} else {analisa kasus }  
               if not IsMember(FirstElmt(H1),H2) then false  
               else { e anggota H2 }  
                   IsSubSet (Tail(H1),H2)
```

```
def is_subset(H1,H2):  
    if is_empty(H1):  
        return True  
    else:  
        if not(is_member(H2,first_element(H1))):  
            return False  
        else:  
            return is_subset(tail(H1),H2)
```

```
L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']
```

```
L2=[2,4,7,'a','x']
```

```
L3=make_set(L1) → L3=['ada', 'b', 'd', 'v', 'a']
```

```
L4=['b','a','v']
```

```
print(is_subset(L4,L3) → True
```

```
print(is_subset(L3,L4) → False
```

Menguji Kesamaan dua SET

KESAMAAN DUA SET	IsEQSet (H1,H2)
<u>DEFINISI PREDIKAT</u>	
IsEQSet : 2 set \rightarrow <u>boolean</u>	
<i>{ IsEQSet (H1,H2) true jika H1 "sama dengan" H2, yaitu jika semua elemen H1 juga merupakan elemen H2, tanpa peduli urutannya }</i>	
<i>{ H1==H2 jika dan hanya jika H1 adalah subset H2 dan H2 adalah subset H1 }</i>	
<u>REALISASI</u>	
IsEQSet (H1, H2) : IsSUBSet (H1, H2) <u>and then</u> IsSUBSet (H2, H1)	

```
def is_eq_set(H1,H2):  
    return is_subset(H1,H2) and is_subset(H2,H1)
```

```
L1=['a','d','v','a','d','ada','a','ada','b','d','v','v','a']
```

```
L2=[2,4,7,'a','x']
```

```
L3=make_set(L1) → L3=['ada', 'b', 'd', 'v', 'a']
```

```
L4=['b','a','v']
```

```
L5=['v','a','b']
```

```
print(is_eq_set(L4,L3))      → False
```

```
print(is_eq_set(L4,L5))      → True
```