1) # max2()

max2 : 2 integer → integer

max2(a,b) menentukan bilangan maksimum dari dua bilangan integer

Realisasi dalam Python

```
def max2(a,b):
    if (a>b):
        return a
    else:
        return b
```

min2()

min2 : 2 integer → integer

min2(a,b) menentukan nilai minimum dari dua bilangan integer

Realisasi dalam Python

```
def min2(a,b):
    if (a<b):
        return a
    else:
        return b
```

Pendefinisian fungsi is_one_element

```
def is_one_element(L):
    return True if ((not empty_list(L)) and empty_list(tail(L))) else False
```

max_list()

max_list : list → integer

max_list(L) menentukan nilai maksimum dari sebuah list berupa integer

Realisasi dalam Python

```
def max_list(L):
    if empty_list(L):
        return 0
    elif is_one_element(L):
        return first_element(L)
    else:
        return max2(first_element(L), max_list(tail(L)))
```

min_list()

min_list : list → integer

min_list(L) menentukan bilangan minimum dari sebuah list

Realisasi dalam Python

```
def min_list(L):
    if empty_list(L):
        return 0
    elif is_one_element(L):
        return first_element(L)
```

```
else :
    return max2 (first_element(L), max_list (tail(L)))
```

2) a total_elemen_daun()

\# total_elemen_daun : pohon Biner → integer

\# total_elemen_daun(P) menghitung jumlah daun pada pohon biner

\# Realisasi dalam python

```
def total_elemen_daun(P) :
    if is_one_elemen(P) :
        return akar(P)
    else :
        if is_biner(P) :
            return total_elemen_daun(left(P)) + total_elemen_daun(right(P))
        elif is_uner_left(P) :
            return total_elemen_daun(left(P))
        elif is_uner_right(P) :
            return total_elemen_daun(right(P))
```

c total_elemen_node()

\# total_elemen_node : pohon Biner → integer

\# total_elemen_node(P) menghitung jumlah semua elemen pada pohon biner

\# Realisasi dalam Python

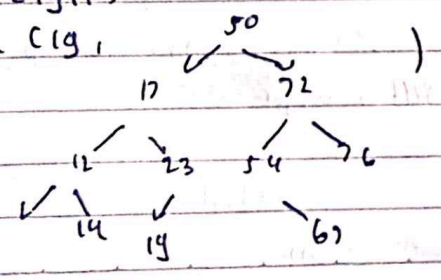
```
def total_elemen_node(P) :
    if is_one_elemen(P) :
        return akar(P)
    else :
        if is_biner(P) :
            return total_elemen_node(left(P)) + akar(P) + totalelemen_node(right(P))
        elif is_uner_left(P) :
            return totalelemen_node(left(P)) + akar(P)
        elif is_uner_right(P) :
            return akar(P) + totalelemen_node(right(P))
```

e BST()

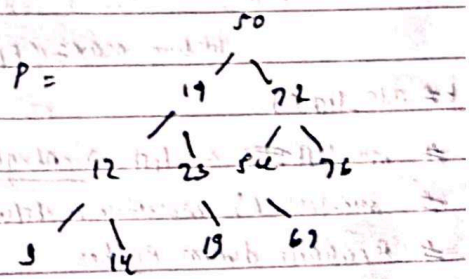
Mencari elemen ig dari pohon biner

search(ig, P)

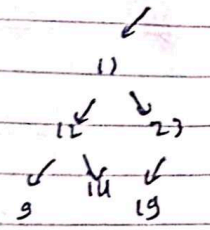
search(ig,



P =

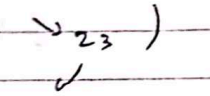


1. karena B < 50 maka yang dicari adalah nilai terbesar pada pohon

search (ig, 

)

2. langkah selanjutnya adalah mengecek apakah ig sama dengan 17 jawabannya false sehingga lanjut membandingkan apakah ig lebih dari 17 ~~jawabannya true~~ karena iya maka ~~dia~~ fokus pencarian berlanjut ke sisi kanan

search (ig, 

)

3. mengecek apakah 23 sama dengan ig jawabannya false karena masih ada node lagi maka dicek lagi ig apakah sama dengan ig maka True sehingga benar elemen ig ada dipohon tersebut.

search (ig, ig) → True

b max_elemen_daun

i. membuat fungsi yang bisa kumpulan list daun-daun pohon biner

ListDaun(P) : PB → list

ListDaun(P) berisi kumpulan list daun dari pohon biner

Program dalam Python

~~def ListDaun(P):~~

~~if isDaun~~

```
def ListDaun(P):
    if is_one_element(P):
        return [Akar(P)]
    else:
        if isBiner(P):
            return ListDaun(leftP) + ListDaun(rightP))
        elif is_unary_left(P):
            return ListDaun(left(P))
        elif is_unary_right(P):
            return ListDaun(right(P))
def max2(a,b):
    if(a > b):
        return a
    else:
        return b
```

GELATIK

```
def max_list(listDown(P)):
    if empty_list(listDown(P)):
        return 0
    elif is_one_element(listDown(P)):
        return First_element(listDown(P))
    else:
        return
```

```
def max_elemen_down(listDown(P)):
    if empty_list(listDown(P)):
        return 0
    elif is_one_element(listDown(P)):
        return FirstElement(listDown(P))
    else:
        return max2(FirstElement(listDown(P)), max_elemen_down(Tail(listDown(L))))
```

* untuk max_elemen_down saya membuat 3 fungsi yaitu menjadikan list kumpulan daun dan membandingkan dengan fungsi max2 yang kemudian ditampilkan ke output

d rata2_elemen_node (a : PB → real

rata2_elemen_node(P) mencari nilai rata-rata dari jumlah elemen dibagi banyaknya elemen

Fungsi sumElmt(P)

```
# sumElmt(P):
    if is_one_elementPB(P):
        return Akar(P)
    else:
        if is_biner(P):
            return sumElmt(Left(P))+akar(P)+sumElmt(Right(P))
        elif is_uner_left(P):
            return sumElmt(Left(P))+akar(P)
        elif is_uner_right(P):
            return akar(P)+sumElmt(Right(P))
```

```
# def NBElmtPB(P)
def NBElmtPB(P):
    if (is_one_elementPB(P)):
        return 1
    else:
        if is_biner(P):
            return NBElmtPB(Left(P))+1+NBElmt(Right(P))
        elif is_uner_left(P):
            return NBElmtPB(Left(P))+1
        elif is_uner_right(P):
            return NBElmtPB(Right(P))+1
```

~~def ratio_element_node(P):~~

~~return sumElmt(P) / NOElmtPB(P)~~

def ratio2_elmn_node(P):

return sumElmt(P) / NOElmtPB(P)

3. a Fungsi dalam Python

```
# kelipatan5 : integer → boolean
# kelipatan5(x) bernilai true jika nilai bilangan bulat x adalah kelipatan 5
# Realisasi dalam Python
def kelipatan5(x):
    if x%5 == 0:
        return True
    else:
        return False
# bukan_kelipatan5 : integer → boolean
# bukan_kelipatan5(x) bernilai true jika bilangan tersebut bukan kelipatan 5
# Realisasi dalam python
def bukan_kelipatan5(x):
    if x%5 != 0:
        return True
    else:
        return False
# Filter_list : List → List
# Filter_list(L) memfilter elemen list yang bernilai kelipatan 5 atau bukan kelipatan 5
# realisasi dalam python
def Filter_list(L,f):
    if empty_list(L):
        return []
    elif f(First_elmt(L)):
        return konso(first_elmt(L), Filter_list(tail(L),f))
    else:
        return Filter_list(tail(L),f)
```

b Ekspresi lambda untuk L2 dan L3

```
L2 = Filter_list(L1, lambda x : x%5 == 0)
L3 = Filter_list(L1, lambda x : x%5 != 0)
```

4)

```
# is_membr : elemen, list → boolean
# is_membr(x,L) bernilai benar jika x adalah elemen dari list L
# Realisasi dalam Python
def isMembr(x,L):
    if (isEmpty(L)):
        return False
    else:
        if FirstElmt(L) == x:
            return True
        else:
            return isMembr(x,Tail(L))
```

is_sub_set : 2 list → boolean
is_sub_set (H1, H2) bernilai True jika semua elemen H1 adalah elemen H2.
Realisasi dalam Python:

```
def is_sub_set (H1, H2):
    if (isEmpty(H1)):
        return True
    elif not is_member(First_Elmt(H1), H2):
        return False
    else:
        return is_sub_set(Tail(H1), H2)
```

minus (A,B)
minus : 2 list → list
minus (A,B) menghasilkan hasil dari selisih dua buah himpunan antara himpunan A dan himpunan B.
Realisasi dalam python

```
def minus (A,B):
    if is_sub_set(A,B):
        return []
    elif is_member(First_Elmt(A), B):
        return minus(Tail(A), B)
    else:
        return Konso(First_Elmt(A), minus(Tail(A), B))
```