Abyan Ardiatama
24060120140161

No
Date

UAS GASAL 2020/2021

1. a) max2 ()

※ max2 : 2integer → integer
※ {max2 (a,b) menentukan bilangan terbesar dari 2 bilangan integer}
※ REALISASI

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

b) min2 ()

※ DEFSPEK
※ min 2 : 2 integer → integer
※ { min2 (a,b) menentukan bilangan terkecil dari 2 bilangan integer}
※ REALISASI

```
def min 2 (a,b) :
    if (a < b) :
        return a
    else :
        return b
```

c) max_list ()

※ DEF~~SPEK~~
※ empty_list : list → boolean {menentukan list kosong atau tidak}
※ first - element : list → integer {menentukan elemen pertama list }
※ tail : list → list {tail(L) menentukan list tanpa elemen pertama }
※ max_list : list → integer
※ { menentukan bil. maksimum dari sebuah list }

※ REALISASI

```
def empty_list (L):
    if (L == []) :
        return True
    else :
        return False
```

No
Date

```
def first_element (L) :
    return L[0]
def tail (L) :
    return L[1:]
def IsOneElmt (L) :
    if (NbElmt==1) :
        return True
    else :
        return False
def max_list (L) :
    if (empty_list (L)) :
        return 0
    elif (IsOneElmt (L)):
        return first_element(L)
    else :
        return max2 (first_element (L), max_list(tail(L))
```

d) min_list ()

* DEFSPEK

* min_list : list → integer

* { min_list(L) menghasilkan bilangan minimum dari sebuah list

* REALISASI

```
def min_list (L) :
    if empty_list(L) :
        return 0
    elif IsOneElmt(L) :
        return first_element(L)
    else :
        return min2 (first_element(L), min_list (tail(L))
```

* APLIKASI

```
L1 = [ 9, 2, 9, 1, 4, -3, 10, -9, 1 ]
print (max_list (L1))
print (min_list (L1))
```

Abyan Ardiatama
24060120140161

No
Date

2. a) total_elemen_daun()

```
* DEFSPEK
* total_elemen_daun : PohonBiner → Integer
* { total_elemen_daun (P) menghitung hasil penjumlahan daun pada pohon biner }

* REALISASI
def total_elemen_daun (P) :
    if (IsOneElmt (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))
        else :
            return total_elemen_daun (right (P))
```

b) total_elemen_node ()

```
* DEFSPEK
* total_elemen_node : PohonBiner → Integer
* { total_elemen_node (P) menghitung hasil penjumlahan semua elemen pada
*   pohonbiner }

* REALISASI
def total_elemen_node (P) :
    if (IsOneElmt (P)) :
        return Akar (P)
    else :
        if (is_biner (P)) :
            return total_elemen_node (left (P)) + Akar (P) + total_elemen_
            node (right (P))
        elif (is_uner_left (P)) :
            return total_elemen_node (left (P)) + Akar (P)
        else :
            return total_elemen_node (right)) + Akar (P)
```

campus

No
Date

✻ APLIKASI
```
P1 = MakePB (8,
          MakePB (3,
                 MakePB(1, None, None), MakePB (6,
                       MakePB (4, None, None), MakePB (7, None, None))),
          MakePB (10, None,
                 MakePB (14,
                       MakePB (13, None, None), None)))
print ( total_elemen_daun (P1))   # hasil = 25
print ( total_elemen_node (P1))   # hasil = 66
```

c) BST()

✻ DEFSPEK

✻ BST [illegible] : PohonBiner, elemen ⟶ boolean

✻ {BST(P,X) bernilai True jika ada node bernilai X pada pohon biner P}

✻ REALISASI
```
def BST(P,X):
    if (IsTreeEmpty (P)):
        return False
    else:
        if (Akar (P) == X):
            return True
        elif (Akar (P) [illegible] X):
            return BST (right (P), X)
        else:
            return BST (left (P), X)
```

✻ APLIKASI
```
print (BST(P1, 7))   # hasil = True
```

Abyan Ardiatama
24060120140161

No
Date

Langkah - langkah

1. Memeriksa apakah P! kosong
2. Jika tidak, node X = 7 dibandingkan denga akar dari pohon P
3. Jika tidak sama, maka akan menjalankan fungsi rekursif BST ke bagian kanan pohon P
4. Apabila di kanan tidak ada node X = 7, maka akan dijalankan fungsi rekursif BST ke bagian kiri pohon P, dan begitu sebaliknya
5. fungsi rekursif tersebut akan terus berjalan hingga elemen pohon P telah habis dibandingkan dengan node X = 7
6. Apabila terdapat node X = 7 sebelum pohon P kosong (empty) akan bernilai True
7. Jika tidak, akan bernilai false

campus

Abyan Ardiatama
24060120140161

No
Date

3. a.) Filter_List() , kelipatan 10() , bukan_kelipatan_10 ()

* DEFSPEK
* Filter_List : list → list
* { menghasilkan list yang telah difilter berdasarkan Filter_List(L,f)
* kelipatan 10 : integer → boolean
* { bernilai benar jika x habis dibagi 10 , kelipatan10 (x) }
* bukan_kelipatan10 : integer → boolean
* { bukan_kelipatan10(x) bernilai true jika x tidak habis dibagi 10 }

* REALISASI
```
def Filter_List (L, f):
    if empty_list (L) :
        return []
    elif (f (first_element(L)) :
        return (Konso (first_element(L), Filter_list ( tail(L), f))
    else :
        return Filter_List (tail (L), f )
  def kelipatan10 (x) :
      return x % 10 == 0
  def bukan_kelipatan_10 (x)
      return x % 10 != 0
```

b) L2 = Filter_List (L1 , lambda x : x % 10 == 0)
* atau
L2 = Filter_List (L1 , kelipatan10)

L3 = Filter_List (L1 , bukan_kelipatan_10)
* atau
L3 = Fiter_List (L1 , lambda x : x % 10 != 0)

* APLIKASI
```
print (L2)
print (L3)
```

campus

No
Date

4 minus ()

✱ DEFSPEK

✱ minus : 2 set ⟶ set

✱ { menghasilkan hasil selisih dari 2 himpunan/set }

✱ REALISASI

```
def is_member (x,L):        ✱ { bernilai true jika x adlh elemen list L }
    if empty_list (L) :
        return False
    else :
        if (first_element (L) == x ) :
            return True
        else :
            return is_member (x, tail (L))

def is_subset (H1,H2):
    if empty_list (H1) :      ✱ { bernilai True jika H1 adlh subset dari H2 }
        return True
    elif not is_member (first_element (H1), H2) :
        return False
    else :
        return is_subset (tail (H1) , H2)

def minus (H1, H2)
    if is_subset (H1, H2) :
        return [ ]
    elif is_member (first_element (H1) , H2) :
        return minus (tail (H1), H2)
    else :
        return konso ([first_element (H1)], minus (tail (H1) , H2))
```

✱ APLIKASI

```
A: [5, 2, 6, 7, 9, 15
B = [2, 7, 15 ]
print (minus (A, B))    ✱ hasil = [ 5, 6, 9 ]
```