PEMBAHASAN UAS DASAR PEMROGRAMAN 2020/2021

```
1. a. max2()
       # max 2: 2 integer - integer
      * Max2 (a.b) menentukan bilangan maksimum dani 2 bilangan integer
      * Realisasi
      def max 2 (a, b):
         If (a>b):
             return a
          else :
             return b
   b. min2()
      # min 2 : 2 integer -> integer
      4 min2(a,b) menentukan bilangan maksimum dari 2 bilangan integer
      def min2 (a,b):
         If (a < b):
           return a
         else:
           return b
  C. Max_list()
     # max_list: list -> integer
    4 max_list (L) Menentukan bilangan maksimum dari sebuah list
 # Realisasi
                                                     untuk no. 10 dan 1d
  def max_list (L):
     If empty_list (L):
                                                     det 15-one - element (L):
       return 0
                                                       If ((not empty_list (L)) and
                                                            empty-list (tall(L))):
    elif 15-one-element (L):
                                                          return true
       return first-element (1)
                                                        else :
       return max 2 (first_element (L), max_list (tail(L))) return false
d. Minlist ()
   # min_list: lise -> wteger
   # 100 min -list (L) merentukan bilangan minimum dan sebuah list
   # Realisari
   def min_list (1):
                                         > else:
    If empty_list (L):
                                              return min2 (First_element (L), min_list
        return O
                                                               (tail(L)))
     elif is - one - element (L):
      return pirst - element (L)
```

DIKLAT HMIF 2020

```
2 a. total_ elemen_down ()
  # total_elemen_daun : Pohon Biner -> integer
  # total_elemen_daun (P) menghitung jumlah daun pada pohon biner
  # Realisasi
  def total - elemen - daun (P):
     If Is_one_element (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 15_unar_right (p):
           return total - elemen-daun (right (p))
 b. total - eleman - node ()
  # total-elemen-node: Pohon Biner -> integer
  # total_elemen- ... (P) menghitung jumlah semua elemen pada pohon biner
  # Realisasi
  def total_elemen_node (P)
     If is_one_alement (P);
        return akar (p)
    else:
        If Is-biner (P):
          return total_elemen_node (left (p)) + akar (p) + total_elemen_node (right (p))
       elif 15 - Uner-left (P):
         return total_elemen_node (left (P)) + akar (P)
     elif is uner right (P):
        return akar(p) + total_elemen _ node (right(p))
C. BST ()
    JIKO P = /LARY
                          1. Cek apakah akar 8 15 biner , tembahiya , cek 'Apakah
                             L < R
                          2. Cek apakah akar 3 Is-biner, teorphilya, cek apakah
                             L<R.
                          3. Cek apakah akar 1 is_biner, ternyata akar 1 tidak
                             ada cabang (daun)
                          4. Cek ke bogion kanan akar 3 , yaitu 6 . Apakah akar 6
                             15-biner. : · ternyata iya dan ak apakah L < R
                         5. Lidin R dan avor 6 ada 4 dan 7 yang
                             tidak memiliki cabang lagi (daun)
```

DIKLAT HMIF 2020 2

	6. setelah dari kiri akar 8, Lalu cek bagian '
: P=[8,3,1,6,4,7,	Kanannya, yaity 10.
10,14,13]	7. Akar 10 hanya memiliki Cabang kanan (uner_right)
	yaity 14.
	8. Akar 14 hanya memiliki Cabang kini (Uner-left)
	yaita 13
The same of the same	9. akar 13 tidak memiliki cabang (daun)
_	. Ta. 4.1 #
3. a. Fungsi pada pytl	The state of the s
# Kelipatan 10 : in	
# Kelipatanio (x) t	pernilai trup bila bilangan merupakan kelipatan 10
# Realisasi	and the second of the sample of a
def kelipatan(o(x)	
lf x % to ==	
refurn True	Andrew Control of the
else :	
return Fals	8
the bulgar halouday	In . Macon 2 hour
	lo : integer -> boolean
H bukan-kelipatan	
H bukan-kelipatan H Realisasi	10 (X) bernilai trup bila bilangan leer bukan merupakan kelipatan 1
H bukan-kelipatan H Realisasi det bukan-kelipat	an 10 (x):
H bukan-kelipatan H Realisasi det bukan-kelipat 1f × % 10!=	an 10 (x):
H bukan-kelipatan H Realisasi det bukan-kelipat 1f × % 10!= return Trve	an 10 (x):
H bukan-kelipatan H Realisasi det bukan-kelipata 1f × % 10!= return True else:	alo(x) bernilai trup bila bilangan leer bukan merupakan kelupatan n an 10 (x):
H bukan-kelipatan H Realisasi det bukan-kelipata 1f × % 10 != return True else: return Falsa	an 10 (x) bernilai trup bila bilangan her bukan merupakan kelupatan 1 an 10 (x):
H bukan-kelipatan H Realisasi def bukan-kelipata 1f × % 10!= return True else: return False # Filter_list: list	10 (X) bernilai trup bila bilangan lett bukan merupakan kelupatan nan10 (X): O
H bukan-kelipatan H Realisasi def bukan-kelipata If x % 10 != return True else: return false # Filter_list: 15st # Filter_list (L) mem	10 (X) bernilai trup bila bilangan lett bukan merupakan kelupatan 1 an 10 (X): 0 1 1/54
H bukan-kelipatan H Realisasi def bukan-kelipata 1f × % 10!= return True else: return False # Filter_list: list	an 10 (x) bernilai trup bila bilangan leer bukan merupakan kelupatan n an 10 (x):
H bukan-kelipatan H Realisasi def bukan-kelipata If x % 10 != return True else: return false # Filter_list: 15st # Filter_list (L) mem	10 (X) bernilai trup bila bilangan lett bukon merupakan kelipatan 1 an 10 (X): 0 1 o 1 list n-filter elemen list yang bernilai kelipatan 10 atau buka
H bukan-kelipatan H Realisasi det bukan-kelipata If x % 10!= return True else: return False # Filter_list: 11st # Filter_list (L) men H Realisasi	10 (X) bernilai trup bila bilangan ker bukan merupakan kelipatan 1 an 10 (X): 0 e -> list n-filter elemen list yang bernilai kelipatan 10 atau buka
# bukan-kelipatan # Realisasi def bukan-kelipata If x % 10 != return True else: return False # Filter_list : list # Filter_list (L) mem # Realisasi def Filter-list (Lif)	10 (X) bernilai trup bila bilangan leer bukan merupakan kelipatan 1 an 10 (X): 0 e -> list n-filter elemen list yang bernilai kelipatan 10 atau buka
# bukan-kelipatan # Realisasi def bukan-kelipata If x % 10!= return True else: return false # Filter_list: 15t # Filter_list (L) mem # Realisasi def Filter_list (Lif) If empty_list (L)	10 (X) bernilai trup bila bilangan ker bukan merupakan kelipatan 1 an 10 (X): 0 e -> list n-filter elemen list yang bernilai kelipatan 10 atau buka :
# bukan-kelipatan # Realisasi det bukan-kelipata If x % 10 != return Trve else: return False # Filter_list: 15st # Filter_list (L) mem # Realisasi det Filter_list (Lif) If empty_list (L) return []	No (x) bernilai trup bila bilangan ker bukan merupakan kelipatan 1 an 10 (x): O P I ist n-filter elemen list yang bernilai kelipatan 10 atau buka : :
# bukan-kelipatan # Realisasi def bukan-kelipata If x % 10!= return True else: return False # Filter_list (L) mem # Realisasi def Filter_list (Lif) If empty_list (L) return [] If not (f (First_e	No (x) bernilai trup bila bilangan lett bukan merupakan kelipatan nan10 (x): O P I ist n-filter elemen list yang bernilai kelipatan 10 atau buka : :

DIKLAT HMIF 2020

```
b. menulis ekspresi Lambda untuk Lz dan L3

L2 = Filter_list (L1, lambda x : x mod 10 = 0)

L3 = Filter_list (L1, lambda x : x mod 10 ≠ 0)

4. Himpunan

# minus : z list → list

# minus (H1, H2) menampilkan himpunan hasil selisih antara himpunan H1 dan H2

# realisasi

def minus (H1, H2):

If Is = sub_set (H1, H2):

return []

elif is_member (First_elm+ (H1), H2):

return minus (tail (H1), H2)

else:

return konso (First_elm+ (H1), minus (tail (H1), H2))
```

DIKLAT HMIF 2020 4