ALPRO UTS 2022/2023

1. Cari hasil akhir jum, dan berapa banyak perintah jum <- jum+(i*j) dieksekusi

```
Kamus
    k, j, jum : integer
Algoritma
    k <- 1; jum <- 0;
    While (k<=3) do
        j traversal[k..3]
            if ((k MOD 2=1) AND (j<=3)) then
                jum <- jum + (k*j)
        {end traversal}
        k <- k+1
    {end while}
```

```
k : 1
  j : 1
    ((k mod 2 = 1 {true}) and (j <= 3 {true})
    jum <- jum + (k*j)
    jum <- 0 + (1*1)
    jum : 1
  j : 2
    (k mod 2 = 1 {true}) and (j <= 3 {true})
    jum <- jum + (k*j)
    jum <- 1 + (1*2)
    jum : 3
  j : 3
    (k mod 2 = 1 {true}) and (j <= 3 {true})
    jum <- jum + (k*j)
    jum <- 3 + (1*3)
    jum : 6
  k <- k+1 {k=2}
k : 2
  j : 2
    (k mod 2 = 1 {false}) and (j <= 3 {true})
```

```
  j : 3
    (k mod 2 = 1 {false}) and (j <= 3 {true})
  k <- k+1 {k=3}
k : 3
  j : 3
    (k mod 2 = 1 {true}) and (j <= 3 {true})
    jum <- jum + (k*j)
    jum <- 6 + (3*3)
    jum : 15
  k <- k+1 {k = 4}
```

Hasil akhir:

jum : 15

Perintah jum <- jum+(k*j) dijalankan sebanyak 4 kali

2. Anagram

Anagram cara 1

```
function isAnagram(T1, T2: array of char, sizeT1, sizeT2 :
integer) -> boolean

  Kamus lokal
    i, j : integer
    cek : boolean
    count : integer

  Algoritma
    count <- 0
    if sizeT1 =/= sizeT2 then
      -> false
    i traversal [1..sizeT1]
      cek <- true
      j traversal [1..sizeT2]
        if (T1_i = T2_j AND cek)then
          count <- count + 1 {increment counter}
          T1_i <- ''
          T2_j <- ''
          cek <- false
      {end traversal j}
    {end traversal i}
    if count = sizeT1 then
      -> true
    -> false
```

Anagram cara 2

```
function toInteger(char X) -> integer

  Kamus lokal
    angka : integer

  Algoritma
    depend on X:
      X = 'A' : angka <- 1
      X = 'B' : angka <- 2
      X = 'C' : angka <- 3
      X = 'D' : angka <- 4
      X = 'E' : angka <- 5
      X = 'F' : angka <- 6
      X = 'G' : angka <- 7
      X = 'H' : angka <- 8
      X = 'I' : angka <- 9
      X = 'J' : angka <- 10
      X = 'K' : angka <- 11
      X = 'L' : angka <- 12
      X = 'M' : angka <- 13
      X = 'N' : angka <- 14
      X = 'O' : angka <- 15
      X = 'P' : angka <- 16
      X = 'Q' : angka <- 17
      X = 'R' : angka <- 18
      X = 'S' : angka <- 19
      X = 'T' : angka <- 20
      X = 'U' : angka <- 21
      X = 'V' : angka <- 22
      X = 'W' : angka <- 23
      X = 'X' : angka <- 24
      X = 'Y' : angka <- 25
      X = 'Z' : angka <- 26
    -> angka
```

```
function isAnagram(T1, T2: array of char, sizeT1, sizeT2 :
integer) -> boolean

  Kamus lokal
    i : integer
    alfabet : array of integers[1..26]

  Algoritma
    i traversal [1..26]
      alfabet_i <- 0

    i traversal [1..sizeT1]
      alfabet_toInteger(T1i) <- alfabet_toInteger(T1i) + 1

    i traversal [1..sizeT2]
      alfabet_toInteger(T2i) <- alfabet_toInteger(T2i) + 1

    i traversal [1..26]
      if alfabet_i =/= 0 then
        -> false
    -> true
```

alfabet = [0, 0, 0, 0, 0, ..., 0]
T1 = “ADE”
T2 = “DEA”

Traversal T1
alfabet = [1, 0, 0, 1, 1, 0, 0, ..., 0]

Traversal T2
alfabet = [0, 0, 0, 0, 0, 0, 0, ..., 0]

Jika semua elemen alfabet = 0, maka Anagram
Jika tidak, bukan Anagram

3.Jumlah array

```
Procedure jumlahArray(input T1, T2 : array of integers, input
sizeT1, sizeT2 : integers, input maxSize : integer, output
hasil : array of integers)

  Kamus Lokal
    Hasil, temp1, temp2 : array of integers[1..maxSize]
    sisa, hasilJuml : integer
    i : integer

  Algoritma
    ReverseArray(T1); ReverseArray(T2);

    i traversal [1..maxSize+1]
      Hasil_i <- 0
      temp1_i <- 0
      temp2_i <- 0
    {end traversal}

    if sizeT1 =/= maxSize+1 then
      i traversal [1..maxSize+1]
        if i <= sizeT1 then
          temp1_i <- T1[i]
        else
          temp1_i <- 0
      {end traversal}
      T1 <- temp1

    if sizeT2 =/= maxSize+1 then
      i traversal [1..maxSize+1]
        if i <= sizeT2 then
          temp2_i <- T1[i]
        else
          temp2_i <- 0
      {end traversal}
      T2 < temp2

    i traversal [1..maxSize+1]
      hasilJuml <- T1_i + T2_i + sisa
      if hasilJuml > 10 then
        sisa <- hasilJuml DIV 10
        hasilJuml <- hasilJuml MOD 10
      else
```

```
        sisa <- 0
      Hasil_i <- hasilJuml
    {end traversal}

    ReverseArray(Hasil)
    hasil <- Hasil
```

T1 = <1 3 4 5 7 8 9>
T2 = <9 3 7 5 7 8>

reverse(T1), reverse(T2)
T1 = <9 8 7 5 4 3 1>
T2 = <8 7 5 7 3 9>

T1 = <9 8 7 5 4 3 1>
T2 = <8 7 5 7 3 9 0>
Hasil <7 6 3 3 8 2 2>

reverse(Hasil)
Haisl = <2 2 8 3 3 6 7>

4.Polindrom

```
Function isPolindrom(T : array[1..100] of integer) -> boolean
  Kamus Lokal
    i : integer
    Ttemp : array [1..100] of integer

  Algoritma
    i traversal [1..100]
      Ttemp_i <- T_i
    {end traversal}

    ReverseArray(Ttemp)
    i traversal [1..100]
      if (Ttemp_i =/= T_i) then
        -> false
    -> true
```

5.