Jawaban hanya sebagai referensi 😊

1. Terdapat kesalahan pada algoritma

```
P <- First(L)
while (Next(P)=/= NIL and (X =/= Info(P)) do
   P <- Next(P)
{ Next(P) = NIL or X = Info(P) }
```

Ketika list L kosong, maka First(L) bernilai NIL. Sehingga nilai awal dari P secara otomatis juga akan bernilai NIL. Selanjutnya, pada perulangan, terdapat pemanggilan Next(P) dan Info(P). Namun, jika P bernilai NIL, maka pemanggilan Next(P) dan Info(P) akan gagal dan menghasilkan error.

Selain itu, pada definisi prosedur, tidak terdapat parameter untuk menyimpan hasil dari prosedur ListSearchFatal. Sehingga prosedur tersebut tidak akan mengubah apa pun.
Procedure ListSearchFatal (input L: List, input X: InfoType)

Masalah ini dapat diselesaikan dengan dua cara:

a. Menambahkan parameter output Found ke dalam definisi prosedur
   Procedure ListSearchFatal (input L: List, input X: InfoType, output Found: Boolean)

b. Mengubah prosedur menjadi sebuah function yang mengembalikan nilai boolean dari hasil kalkulasi algoritma
   Function ListSearchFatal (L: List, X: InfoType) -> Boolean

2.

```
Function IsListSimetri (L1: List, L2: List) -> boolean
{Mengirimkan TRUE jika setiap elemen pada L1 sama dengan L2
(Simetri)}
{Ex. Jika L1: 1->2->3->5, L2: 1->2->3->6, maka FALSE}
{    Jika L1: 1->2->3->5, L2: 1->2->3->5, maka TRUE}

{Kamus Lokal}
  P1, P2 : Address
  found : Boolean

{Algoritma}
  if (NbElmt(L1) = NbElmt(L2) then
    if IsEmpty(L1) then
      -> True
    P1 <- First(L1)
    P2 <- First(L2)
    while (Info(P1) = Info(P2) and Next(P1) =/= First(L1)) do
      P1 <- Next(P1)
      P2 <- Next(P2)
    { Info(P1) =/= Info(P2) or Next(P1) = First(L1) }
    if Info(P1) =/= Info(P2) then
      -> False
    -> True
  else
    -> False
```

Jawaban hanya sebagai referensi 😊

3.

```
Procedure IntersectList(input:L1:List, input:L2:List,
output:L3:List)

{Kamus Lokal}
  P1, P2 : Address

{Algoritma}
  if NbElmt(L1) > 0 and NbElmt(L2) > 0 then
    P1 <- First(L1)
    while (P1 =/= NIL) do
      P2 <- First(L2)
      while (Info(P1) =/= Info(P2) and P2 =/= NIL) do
        P2 <- Next(P2)
      {Info(P1) = Info(P2) or P2 = NIL}
      if Info(P2) = Info(P2) then
        InsertVLast(L3, Info(P2))
      P1 <- Next(P1)
    {P1 = NIL}
```

4.

```
Function IsSimilarTree(P1: BinTree, P2: BinTree) -> Boolean

{Kamus Lokal}

{Algoritma}
  if NbElm(P1) == NbElm(P2) then
    if NbElm(P1) > 0 then
      if akar(P1) = akar(P2) then
        -> IsSimilarTree(left(P1), left(P2)) and
IsSimilarTree(right(P1), right(P2))
      else
        -> False
    else
      -> True
  else
    -> False
```

Jawaban hanya sebagai referensi 😊

Jawaban hanya sebagai referensi 😊

5.

```
Function MaxNode(P: Bintree) -> Integer

{Kamus Lokal}

{Algoritma}
  if IsEmpty(P) then
    -> 0
  else
    if IsDaun(P) then
      -> akar(P)
    else
      -> Max(Max(MaxNode(left(P)), MaxNode(right(P)), akar(P))
```