1) Definisi & Spesifikasi

isBinary Tree : List → Boolean

{ isBinery Tree (L) menghasilkan True jika jumlah elemen pada L lebih besar sama dengan 3 }

Realisasi

isBineryTree (L):

if (isEmpty (L) or (isEmpty (Tail (L)) and Head < 3) then false
else if (Head(L) >= 3) then true
else isBineryTree (Konso (Head (L) + Head (Tail (L)), Tail (Tail (L))))

Aplikasi :

is Binery Tree ([1,2,3]) → True
isBinery Tree ([1,1]) → False
isBinery Tree ([10,2]) → True

2) Definisi & Spesifikasi

GantiXC : Integer, karakter, List → List

{ GantiXC (IX, C, L) menghasilkan list karakter dengan elemen pada posisi IX diganti dengan karakter C }

Realisasi

GantiXC (IX, C, L):

if (IX = 1) then Konso (C, Tail (L))
else Konso (Head (L), GantiXC (IX - 1, C, Tail (L)))

Aplikasi

GantiXC (2, 'F', ['A','B','C']) → ['A','F','C']
GantiXC (1, 'Z', ['A','B','C']) → ['Z','B','C']

3) Definisi & Spesifikasi

MakeDifference : 2 set → set

{ MakeDifference (H1, H2) menghasilkan set baru dengan anggota H1 yang tidak menjadi anggota H2 }

Realisasi

```
MakeDifference (H1,H2):
  depend on H1,H2 :
    IsEmpty(H1) and IsEmpty(H2) : H1
    not IsEmpty(H1) and IsEmpty(H2): H1
    IsEmpty(H1) and not IsEmpty(H2): H1
    not IsEmpty(H1) and not IsEmpty(H2):
      if IsMember (FirstElmt(H1), H2)
      then MakeDifference (Tail(H1), H2)
      else Konso (FirstElmt(H1), MakeDifference (Tail(H1), H2)
```

Aplikasi:

MakeDifference ([1,2,3,4], [2,4,6]) → [1,3]
MakeDifference ([2,3,5,7], [1,4,6]) → [2,3,5,7]

4) Reverse Tree : Pohon Biner tidak kosong → Pohon Biner
{ReverseTree (P) menghasilkan sebuah pohon biner yang dipertukarkan antara sub pohon kiri dengan kanan}

Realisasi

```
ReverseTree (P):
  if IsBiner (P) then
    Konso (Akar (P), Konso (ReverseTree (Right (P)), Konso (ReverseTree (Left (P)), [])))
  else if IsUnerLeft (P) then
    Konso (Akar (P), Konso ([], Konso (ReverseTree (Left (P)), [])))
  else if IsUnerRight (P) then
    Konso (Akar (P), Konso (ReverseTree (Right (P)), Konso ([], [])))
  else P
```

5) Definisi & Spesifikasi
AddDaun Terkanan : pohon biner, elemen → pohon biner
{AddDaun Terkanan (P, X) menghasilkan pohon biner yang ditambahkan elemen X sebagai daun terkanan}

Realisasi

```
AddDaunTerkanan (P,X):
  if not IsExistRight (P) then
    Konso (Akar (P), Konso (Left (P), Konso (Konso (X, Konso ([], Konso ([],[]))), [])))
  else Konso (Akar (P), Konso (Left (P), Konso (AddDaunTerkanan (Right (P), X), [])))
```