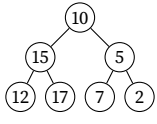
Level-order

En *level-order*-traversering besøker nodene i et tre etter *dybde*. Det vil si at roten besøkes først, så alle barna til roten, så alle barna til rotens barn, og så videre.

En *level-order*-traversering av følgende tre kan, for eksempel, besøke nodene i rekkefølgen $10, 15, 5, 12, 17, 7, 2$.



(a) Anta at treet er en *binær heap*, representert ved et array A med n elementer. Gi en algoritme som returnerer en liste med elementene fra A i *level-order*.

```
Input: En binær heap, representert ved et array A med n elementer
Output: En liste med elementene i heapen i level-order
Procedure LevelOrderHeap(A)
    // ···
```

(b) Anta at treet er et *binært søketre* med n noder, der hver node v har følgende felter:

- v.`element` — verdien som er lagret i noden
- v.`left` — venstre barn av v
- v.`right` — høyre barn av v

Gi en algoritme som returnerer en liste med elementene fra treet i *level-order*.

```
Input: Roten v av et binært søketre med n noder
Output: En liste med elementene i søketreet i level-order
Procedure LevelOrderBST(v)
    // ···
```

(c) Oppgi kjøretidskompleksiteten for algoritmene dine i (a) og (b).

(d) Se på følgende algoritme:

```
Input: Roten v av et binært søketre
Output: Et nytt binært søketre
Procedure ConstructBST(v)
    B ← empty binary search tree
    nodes ← LevelOrderBST(v)
    for u in nodes do
        B ← Insert(B, u)
    return B
```

Hva kan du si om relasjonen mellom input-treet og output-treet?