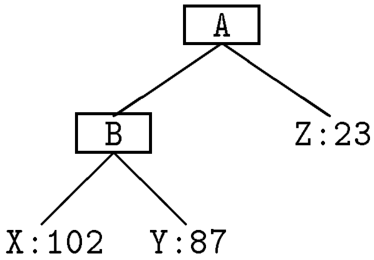
Opgave 1 (20%)

Betragt følgende type, der skitserer et UNIX filsystem:

> **Type** Directory = **Prod**(name: Text, dirs: DirList, files: FileList)
> **Type** File = **Prod**(name: Text, size: Int)
> **Type** DirList = **List**(Directory)
> **Type** FileList = **List**(File)

Vi ønsker en kommando, der udskriver information om filsystemet på følgende måde. For filsystemet:



skal den udskrive denne oversigt:

```
directory A containing
  directory B containing
    file X of 102 bytes
    file Y of 87 bytes
  total 189 bytes
  file Z of 23 bytes
total 212 bytes
```

Skriv en TRINE procedure:

> **Proc** Info[D: Directory]

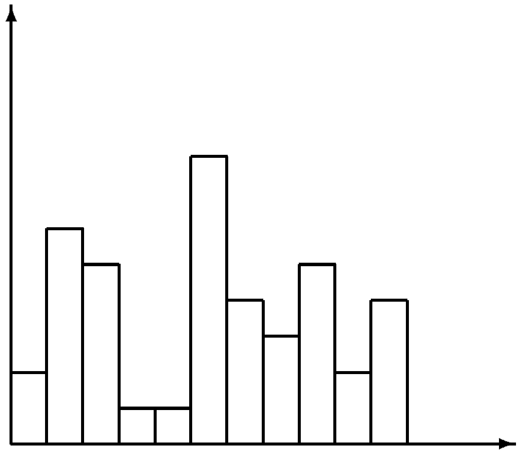
der udskriver en sådan oversigt (med den rigtige indrykning). Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Et *histogram* er en værdi af type:

Type Histo = **List** (Real)

der kan skitseres som følger:



Det *vægtede gennemsnit* af et ikke tomt histogram H er defineret som:

$$vg(H) = \frac{1}{|H|} \sum_{i=0}^{|H|-1} (i+1)*H.(i)$$

Det følgende er en skitse til en algoritme, der beregner og udskriver det vægtede gennemsnit af histogrammet H:

Algoritme: Vægtet gennemsnit
Stimulans: H: |H|>0
Respons: *vg*(H)
Metode: ≪initialiser vg og i≫
do { I }
  i < |H| →
    ≪iterer≫
    i := i+1
od
write(vg)

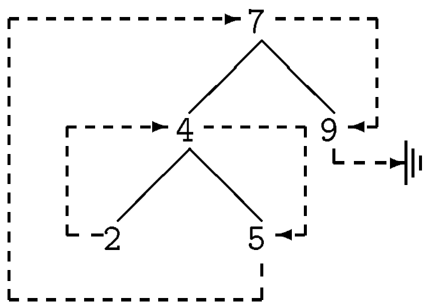
Udfyld ≪initialiser g og i≫ og ≪iterer≫, så algoritmen bliver korrekt. Find en passende invariant og bevis gyldighed, korrekthed og terminering. Der må ikke introduceres andre variabler end vg og i.

Opgave 3 (20%)

Vi skal lave en udvidelse af ordbøger, der understøtter en ekstra operation Sweep. Hvis elementerne fra ordbogen D i voksende orden er $(d_0, d_2, \ldots, d_{n-1})$ og $x = d_i$, så skal `Sweep[D](x, k)` returnere en liste med de k efterfølgere til x, det vil sige $(d_{i+1}, \ldots, d_{i+k})$. Hvis $n - 1 < i + k$ så stopper listen ved d_{n-1}.

a) Giv en formel specifikation af proceduren Sweep.

Vi ønsker at implementere den udvidede ordbog, så også Sweep bliver effektiv. Til det formål kan vi benytte et balanceret søgetræ, hvor hver knude yderligere har en pointer til knuden med dets umiddelbare efterfølger i elementordenen, som fx:



hvor de ekstra pointere er stiplede.

b) Angiv hvorledes dette udvidede søgetræ kan benyttes, så `Init[D]` får tidskompleksitet $O(1)$, `Insert[D](x)`, `Member[D](x)` og `Delete[D](x)` får tidskompleksitet $O(\log \|D\|)$, og `Sweep[D](x, k)` får tidskompleksitet $O(\log \|D\| + k)$. Du skal for hver operation forklare, hvordan den benytter og bevarer invarianten.

Opgave 4 (20%)

Betragt følgende TRINE program.

```
(+ Box B(S)
     Proc P[s:S]
       (*1*)
       s:=s+1
     end P
   end B

   Box A
     B(Int)
   end A

   Type T = List(Prod(x: Bool, y: Sum(a: Int, b: Real), z: T))

   Var x: T

   A'P[x.(8).z.(7).y.a]
+)
```

a) Angiv normalformen af typen T.

b) Hvad er den statiske omgivelse for stedet (*1*)?

c) Vil programmet blive accepteret af TRINE oversætteren? Begrund dit svar.

Opgave 5 (20%)

Følgende program fra Grafalgoritmer afsnit 3.1 kan benyttes til farvning af ikke-orienterede grafer.

```
Algoritme: Graffarvning
Stimulans:  G = (V, E) ikke orienteret "hvid" graf
Respons  :  G, hvor alle knuder og kanter er farvet røde
Metode   :  NS'Init[PLR](n)
            MS'Init[H](n)
            do ¬ (MS'Empty[H] ∧ NS'Empty[PLR]) →
                if NS'Empty[PLR] →
                    MS'DeleteSome[H, v]
                    ≪farv v rød≫
                    NS'Insert[PLR](v)
                & true →
                    NS'DeleteSome[PLR, v]
                    for (v, w) in E do
                        if NS'Member[PLR](w) →
                            ≪farv (v, w) rød≫
                         | MS'Member[H](w) →
                            MS'Delete[H](w)
                            ≪farv (v, w) og w rød≫
                            NS'Insert[PLR](w)
                        & true → skip
                        fi
                       od
                fi
            od
```

Ved besvarelse af opgaven kan det uden bevis benyttes, at et ***træ*** er en sammenhængende ikke-orienteret graf uden cykler.

Modificér algoritmen, så den i en variabel af type Bool fortæller om grafen G er et træ. Argumentér for korrektheden.