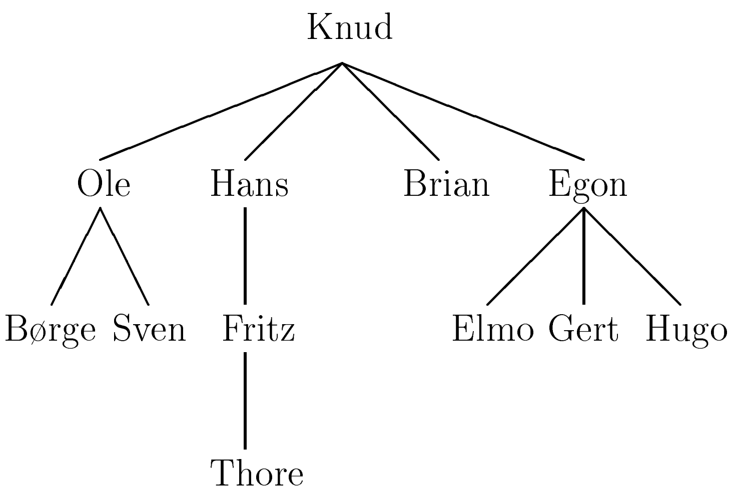
Opgave 1 (20%)

Et mandligt familietræ kan repræsenteres som en værdi af følgende type:

Type Familie = **Prod**(navn: Text, sønner: **List**(Familie))

Vi antager i denne opgave, at alle familiemedlemmer har indbyrdes forskellige navne.

Din *fætter* er som bekendt en søn af en af din fars brødre. I familietræet:



er Børges fætre således Fritz, Elmo, Gert og Hugo (Sven er jo hans bror, og Thore er hans nevø).

Skriv en procedure:

Proc Fætre[F: Familie](n: Text)

der udskriver navnene på alle n's fætre i familietræet F (det kan antages, at n vides at befinde sig i træet). Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Det er velkendt, at følgende algoritme er gyldig og korrekt.

Algoritme: Division
Stimulans: a,b: $(a \geq 0) \wedge (b > 0)$
Respons: q,r: $(a = q*b+r) \wedge (0 \leq r < b)$
Metode:

```
q, r := 0, a
do { (a = q*b+r) ∧ (0 ≤ r) }
    r ≥ b → q, r := q+1, r-b
od
```

Betragt nu følgende alternative version af algoritmen, hvor vi anvender operatoren $\uparrow$ til at angive multiplikation med 10-potenser; mere præcist gælder der for alle heltal h,p≥0:

$$h \uparrow p = h*10^{p}$$

Algoritme: Alternativ Division
Stimulans: a,b: $(a \geq 0) \wedge (b > 0)$
Respons: q,r: $(a = q*b+r) \wedge (0 \leq r < b)$
Metode:

```
p := 0
do b↑p ≤ a → p := p+1 od
q, r := 0, a
do { (a = q*b+r) ∧ (0 ≤ r<b↑p) ∧ (0 ≤ p) }
    p>0 → p := p-1
            do { (a = q*b+r) ∧ (0 ≤ r) ∧ (0 ≤ p) }
                r ≥ b↑p → q, r := q+(1↑p), r-(b↑p)
            od
od
```

a) Bevis, at denne algoritme er gyldig og korrekt.

b) Hvad er tidskompleksiteterne for de to algoritmer?

Opgave 3 (20%)

I en vilkårlig orienteret vægtet graf med n knuder og m kanter kan man som bekendt finde længden af den korteste vej mellem to givne knuder i tid $O((n + m) \log n)$ ved brug af Dijkstras algoritme.

I denne opgave skal vi se, hvordan man kan udnytte egenskaberne ved nogle specielle grafer til at opnå forbedrede tidskompleksiteter.

a) Antag, at vi kun ser på grafer, i hvilke alle kanter har vægten 1. Skitser en simpel og effektiv algoritme, der beregner længden af den korteste vej mellem to givne knuder. Argumentér for algoritmens tidskompleksitet.

b) Antag, at vi kun ser på grafer, i hvilke alle kanter har vægten 1 eller 2. Skitser en simpel og effektiv algoritme, der beregner længden af den korteste vej mellem to givne knuder. Argumentér for algoritmens tidskompleksitet.

c) Antag, at vi kun ser på grafer, i hvilke alle knuder har udgrad 0 eller 1. Skitser en simpel og effektiv algoritme, der beregner længden af den korteste vej mellem to givne knuder. Argumentér for algoritmens tidskompleksitet.

Opgave 4 (20%)

I denne opgave betragtes multiplikation af (lange) heltal i forbindelse med evaluering af polynomier. I det følgende er x et heltal med k cifre.

a) Hvor mange cifre har x^n $(n \geq 1)$?

Det vides, at to m-cifrede tal kan multipliceres i tid $O(m^{\log_2 3})$; vi antager mere generelt, at m-cifrede tal kan multipliceres i tid $O(m^\alpha)$, hvor $1 < \alpha$.

b) Angiv, hvordan x^n kan udregnes i tid $O((kn)^\alpha)$.

Hvis tallene ikke er lige lange, men man ganger et m-cifret tal med et der er s gange længere, så kan det gøres ved hjælp af s simplere m-cifrede multiplikationer (og nogle billige additioner) i samlet tid $O(s \cdot m^\alpha)$.

Betragt nu følgende polynomium:

$$p(x) = a_0 + a_1 x + \cdots + a_i x^i + \cdots + a_n x^n$$

hvor $a_n \neq 0$. Man kan evaluere $p(x)$ ved hjælp af Horners metode:

$$p(x) = (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + \cdots + a_1)x + a_0$$

Tiden for denne beregning kan findes som følger. Den første multiplikation $(a_n x)$ tager tid k^α; den anden involverer tal med $2k$ og k cifre og tager tid $2 \cdot k^\alpha$; den i'te involverer tal med ik og k cifre og tager tid $i \cdot k^\alpha$. Hele beregningen tager dermed tid:

$$O(\sum_{i=1}^{n} i \cdot k^\alpha) = O(k^\alpha \sum_{i=1}^{n} i) = O(k^\alpha n^2)$$

c) Angiv, hvordan man kan anvende del-og-kombiner teknikken til at evaluere $p(x)$ (vink: opdel i to beregninger hver af størrelse $\frac{n}{2}$). Hvad bliver tidskompleksiteten?

d) For hvilke værdier af α er tidskompleksiteten i c) bedre end tidskompleksiteten af Horners metode?

Opgave 5 (20%)

En database holder rede på en mængde målinger af nogle menneskers højder. Hvert element i databasen er af formen:

(cpr-nummer,højde,alder)

og registrerer den målte højde for en given person i en given alder. Databasen kan indeholde flere målinger for en given person, men højst en enkelt måling for en given person i en given alder.

Databasen skal realiseres ved hjælp af en box med følgende udseende:

Box Database
  Type DB = ≪mængde af målinger≫
  Proc Init [d: DB]
  Proc Insert [d: DB] (cpr, height, age: Int)
  Proc Delete [d: DB] (cpr: Int)
  Proc CurrentHeight [d: DB] (cpr: Int) → (Int)
  Proc Tallest [d: DB] → (Int)
  Proc Population [d: DB] (age: Int) → (Int)
end Database

Init skaber en tom database. Insert indsætter en ny måling eller opdaterer en eksisterende. Delete sletter alle oplysninger om den givne person. CurrentHeight returnerer den seneste målte højde af den givne person. Tallest returnerer den største højde, der nogensinde er målt på en person i databasen. Population returnerer antallet af personer, der har fået foretaget målinger i den angivne alder.

a) Giv en formel specifikation af operationen CurrentHeight.

I det følgende antager vi, at en alder er et heltal i intervallet [0..150] og at en højdemåling er et heltal i intervallet [0..250]. Lad |d| angive antallet af personer i databasen d.

b) Angiv en implementation af typen DB, så:

- Init `[d]` får tidskompleksitet i $O(1)$,
- Insert `[d]` `(c,h,a)` og Delete `[d]` `(c)` får tidskompleksiteter i $O(\log |d|)$,
- CurrentHeight `[d]` `(c)` får tidskompleksitet i $O(\log |d|)$,
- Tallest `[d]` får tidskompleksitet i $O(1)$,
- og Population `[d]` `(a)` får tidskompleksitet i $O(1)$.