Opgave 1 (20%)

Et HTML_{00} fragment er en værdi af følgende type:

Type Entity = **Sum** (word: Text,
anchor: **Prod** (f: URL, h: HTML_{00}),
bold: HTML_{00})
Type HTML_{00} = **List** (Entity)
Type URL = Text

Et korrekt udformet fragment skal tilfredsstille det yderligere krav, at man ikke må definere et anker ("anchor") inden i et andet anker. Det vil sige, at fx fragmentet:

```
<a href="abc.html">Dette er <a href="xyz.html">forbudt</a></a>
```

der svarer til værdien:

```
HTML00(
   Entity(anchor: Prod("abc.html",
                      HTML00(Entity(word: "Dette"),
                             Entity(word: "er"),
                             Entity(anchor: Prod("xyz.html",
                                                 HTML00(Entity(word: "forbudt"))
                                                )
                                   )
                            )
  ))
)
```

ikke er lovligt. Bemærk, at det indre anker kan stå inde i vilkårligt indviklede HTML_{00}-konstruktioner.

> Skriv en værdiprocedure:
>
> **Proc** NoNestedAnchors [H: HTML_{00}] $\rightarrow$ (Bool)
>
> der afgør, om fragmentet H indeholder et anker inden i et andet anker. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Givet en ikke-tom liste af heltal, S, er vi interesserede i at opbygge en liste af sandhedsværdier, D, således at D.(j) fortæller, om S.(j) er større end eller lig med alle tal til venstre i listen. For listen:

$$S : (2, 5, 3, 7, 4, 13, 13, 5, 11)$$

vil svaret således være (pr. definition er D.(0)=true):

$$D : (\text{true},\text{true},\text{false},\text{true},\text{false},\text{true},\text{true},\text{false},\text{false})$$

Følgende algoritme løser dette problem:

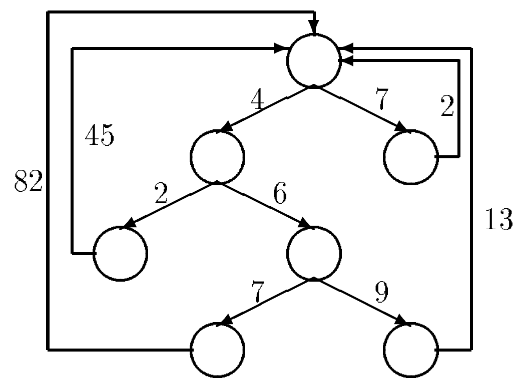
Algoritme: Venstre Profil
Stimulans: S: **List**(Int), $|S| \geq 1$
Respons: D: **List**(Bool), $|D|=|S| \wedge \forall j \in 0..|S|$: $D.(j) \Leftrightarrow (\forall k \in 0..j$: $S.(k) \leq S.(j))$
Metode: D:=**List**(?-Bool | | S |)
i, m, D.(0) := 1, S.(0), true
do { I }
  i< | S | →
   if m ≤ S.(i) → m, D.(i) := S.(i), true
   & true → D.(i) := false
   fi
   i := i+1
od

a) Bevis terminering af algoritmen.

b) Bevis korrekthed af algoritmen, ved blandt andet at angive en gyldig invariant.

Opgave 3 (20%)

Et *rodet træ* er en orienteret vægtet graf med positive vægte, der har form som et binært træ, hvor hvert blad har en kant tilbage til roden:



a) Bevis, at der højst er $2n$ kanter i et rodet træ med n knuder.

Der skal konstrueres en algoritme, der givet to knuder i et rodet træ finder længden af den korteste vej imellem disse.

b) Angiv tidskompleksiteten der garanteres, hvis vi benytter Dijkstras algoritme til at løse dette problem.

c) Beskriv en hurtigere algoritme, der udnytter de specielle egenskaber ved et rodet træ. Angiv den forbedrede tidskompleksitet.

Opgave 4 (20%)

I denne opgave indfører vi en ny binær operator i RASMUS. Syntaksen er:

```
R <a> S
```

Det kræves, at de fælles attributter for R og S har de samme typer og ikke omfatter attributnavnet a. Resultatet af operatoren er værdien af udtrykket:

```
!(R,S): rel(#)*rel(tup(a: |@(1)*@(2)|))
```

Det bemærkes, at udtrykket:

```
!(R,S): U
```

er en forkortelse for udtrykket:

!(R,S)|$n_1,\ldots,n_k$: U

hvor $n_1,\ldots,n_k$ er de fælles attributnavne for R og S. Lad i det følgende Salg og Køb være relationerne:

Sælger:Text	Vare:Text
Fido A/S	hund
Katte Discount	kat
Vov & Co.	hund
Turtles 'R Us	skildpadde
Furry Friends	hamster
Jysk Hamster Import	hamster
Fibonacci Int'l	kanin

og

Vare:Text	Køber:Text
hund	Anders Husfeldt
kat	Thore Hougaard
skildpadde	Jesper Bækgaard
hamster	Ole Pedersen
hamster	Jakob Sandholm
kanin	Jan Lauridsen
guldfisk	Ole Pagter

a) Angiv for eksempelrelationerne resultatet af: Salg <Antal> Køb og giv en intuitiv forklaring på betydningen af dette udtryk.

b) Angiv, hvilke af følgende regneregler, der er gyldige (under antagelse af at begge sider er lovlige relationsudtryk). Begrund dine svar.

1) R <a> R = R

2) R <a> S = S <a> R

3) R <a> (S + T) = (R <a> S) + (R <a> T)

Opgave 5 (20%)

I denne opgave skal der udregnes den mest profitable plan for at investere et antal kroner i en række virksomheder.

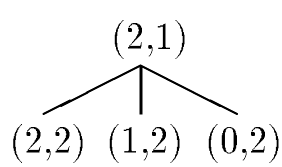
Der er givet virksomhederne $V_1, V_2, ..., V_n$ og en sum på m kroner. Antag endvidere eksistensen af en værdiprocedure Udbytte, for hvilken kaldet Udbytte(k,i) i konstant tid beregner det forventede udbytte af at investere k kroner i virksomheden V_i.

For den følgende værdiprocedure gælder det, at et kald af Invester(m,1) vil returnere det maksimale forventede udbytte af at investere i alt m kroner i de n virksomheder:

```
Proc Invester(k, i: Int) → (Int)
  if i > n → return 0 fi
  (+ Var j, inv: Int
    j, inv := 0, 0
    do j≤k →
        inv := max(inv, Udbytte(j, i)+Invester(k-j, i+1))
        j := j+1
    od
  +)
  return inv
end Invester
```

a) Forklar kort, hvad Invester(k,i) beregner.

Hvis n=2, så ser rekursionstræet for kaldet Invester(2,1) ud som:



b) Antag, at n=4. Tegn rekursionstræet for kaldet Invester(2,1).

På baggrund af en analyse af rekursionstræet for kaldet Invester(k,i) kan det vises, at hvis k>n−i, så er $\mathbf{T}[\![\text{Invester(k,i)}]\!] \in \Omega(2^{n-i})$.

c) Benyt dynamisk programmering til at konstruere en mere effektiv algoritme. Hvad bliver den forbedrede tidskompleksitet?