Opgave 1 (15%)

Et *videnstræ* er (jfr. Dat1 nr. 2 opgave U61) af TRINE typen

> **Type** Vid = **Sum**(ting: Text, spørgsmål: Spørg)
> **Type** Spørg = **Prod**(hvad: Text, ja, nej: Vid)

Videnstræet er ***inkonsistent*** hvis to forskellige blade indeholder samme ting (for så må der jo findes et spørgsmål, hvor man har svaret både *ja* og *nej* for den samme ting).

Det antages i det følgende, at nedenstående box er til rådighed.

```
Box T
    Type Set = ≪mængde af tekster≫

    Proc Single(t: Text) → (Set)
        return ≪{t}≫
    end Single

    Proc Intersect(s1, s2: Set) → (Set)
        return ≪s1 ∩ s2≫
    end Intersect

    Proc Union(s1, s2: Set) → (Set)
        return ≪s1 ∪ s2≫
    end Union
end T
```

Skriv en TRINE værdiprocedure

> **Proc** Ink[V: Vid] → (T'Set)

der beregner mængden af ting, der forekommer to eller flere gange i videnstræet V. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

En bitstreng

$$B = 011000101$$

har en tilhørende ***skiftenøgle***

$$S = (9, (1, 3, 6, 7, 8))$$

der består af B's længde, samt de positioner (nummereret fra 0 til $|B| \Leftrightarrow 1$) for hvilke B skifter fra 0 til 1 eller omvendt. Det vil sige, hvis vi betegner B's skiftenøgle med ***skift***(B), så gælder det, at

$$\textbf{\textit{skift}}(B) = (|B|, \{i \mid 0 < i < |B| : B.(i \Leftrightarrow 1) \neq B.(i)\})$$

Lad S være af typen **Prod**(l: Int, s: Vector) og betragt følgende algoritmeskitse.

Algoritme: Beregn Skiftenøgle

Stimulans: B: Vektor, $(|B| > 0) \wedge \forall i \in 0..|B| : B.(i) \in \{0, 1\})$

Respons: $S = \textbf{\textit{skift}}(B)$

Metode: $\ll$initialiser i og $S\gg$

do $\{S = \textbf{\textit{skift}}(B(0..i)) \wedge (0 < i \leq |B|)\}$

$i \neq |B| \rightarrow \ll$opdater i og $S\gg$

od

a) Gør algoritmen færdig og bevis, at den er korrekt.

b) Det er klart, at ***skift***(B) er entydigt bestemt af B. Gælder det også, at B er entydigt bestemt af ***skift***(B)? Begrund dit svar.

Opgave 3 (15%)

Der skal konstrueres en box SparseList med følgende udseende

> **Box** SparseList
>   **Type** L = ≪uendelig liste af heltal≫
>   **Proc** Init[x: L]
>   **Proc** Lookup[x: L](i: Int) → (Int)
>   **Proc** Update[x: L](i, k: Int)
>   **Proc** Max[x: L] → (Int)
>   **Proc** Add[x: L] → (Int)
> **end** SparseList

som realiserer en datastruktur, hvis værdier er uendelige lister af heltal.

Proceduren Init giver den uendelige liste med nuller på alle pladser. Proceduren Lookup returnerer værdien med index i. Proceduren Update ændrer værdien med index i til at være k. Proceduren Max giver det største ***index***, hvis værdi ikke er nul. Proceduren Add giver summen af listens elementer.

I det følgende angiver $||x||$ antallet af elementer i x, der har værdi forskellig fra nul.

a) Giv en formel specifikation af proceduren Max.

b) Beskriv en realisation af typen L, så Init får tidskompleksitet $O(1)$, Lookup og Update får tidskompleksitet $O(\log ||x||)$, og Max og Add får tidskompleksitet $O(1)$.

Opgave 4 (15%)

Betragt følgende TRINE program.

```
(+ Type A = List(B)
   Type B = Sum(x: C, y: D)
   Type C = C
   Type D = E
   Type E = List(Sum(x: Unit, y: A))

   Var a: A
   Var b: B
   Var c: C
   Var d: D
   Var e: E
   Var i: Int
   Var r: Real

   a := e
   b := Sum(2: List())
   c := ?-C
   r := i
+)
```

Vil det blive accepteret af TRINE oversætteren? Begrund dit svar.

Opgave 5 (20%)

Følgende algoritme fra [Grafalgoritmer] side 39 beregner en topologisk sortering af en orienteret acyklisk graf.

Algoritme: Topologisk Sortering
Stimulans : $G = (V, E)$ orienteret acyklisk graf
Respons : TopSort: Vector, indeholder topologisk sortering af G
Metode :

```
Indegree := Vector(0|n)
for (v, w) in E do
        Indegree.(w) := Indegree.(w)+1
od
NS'Init[R](n)
for v in V do
      if Indegree.(v) = 0 → NS'Insert[R](v) fi
od
TopSort, N := Vector(0|n), 1
do ¬ NS'Empty[R] →
      NS'DeleteSome[R, v]
      TopSort.(v), N := N, N+1
      for (v, w) in E do
            Indegree.(w) := Indegree.(w)-1
            if Indegree.(w) = 0 → NS'Insert[R](w) fi
      od
od
```

Modificér metoden, så den i stedet realiserer følgende algoritme.

Algoritme: Cyklisten
Stimulans : $G = (V, E)$, orienteret graf
Respons : AC: Boolean, AC $\Leftrightarrow$ G er acyklisk
Metode :

Opgave 6 (15%)

Følgende to algoritmiske problemer har oplagte løsninger med udførelsestid i $O(n)$.

P1 Lad $A(0..n)$ være en liste for hvilken $A.(0) \geq A.(1)$ og $A.(n \Leftrightarrow 2) \leq A.(n \Leftrightarrow 1)$. Find et ***lokalt minimum*** i A, det vil sige, et indeks x for hvilket $A.(x \Leftrightarrow 1) \geq A.(x)$ og $A.(x) \leq A.(x+1)$.

P2 Lad $f : N_0 \rightarrow Z$ være en monotont aftagende funktion. Find det mindste n for hvilket $f(n) \leq 0$.

Begge problemer kan imidlertid løses mere effektivt.

> Beskriv sådanne effektive algoritmer for **P1** og **P2** og angiv deres udførelsestider.