Opgave 1 (20%)

En delmængde af relationsudtrykkene fra RASMUS kan repræsenteres som værdier af følgende type:

Type Expr = **Sum**(relation: Text,
project: **Prod**(q: Expr, a: Text),
union, join: **Prod**(q_1, q_2: Expr))

Fx vil udtrykket:

`((R+S)|+x)*T`

blive repræsenteret som værdien:

```
Expr(join:Prod(Expr(project:Prod(Expr(union:Prod(Expr(relation:"R"),
                                                 Expr(relation:"S"))),
                                 "x")),
               Expr(relation:"T")))
```

Vi er interesserede i at optimere sådanne udtryk ved konsekvent at omskrive en project af en union til den ækvivalente union af projects. Fx ønsker vi, at det ovenstående udtryk omskrives til (det mere effektive):

`((R|+x)+(S|+x))*T`

> Skriv en værdiprocedure:
>
> **Proc** Optimize[Q: Expr] → (Expr)
>
> der foretager denne omskrivning på et relationsudtryk af type Expr. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Givet en ikke-tom liste af ikke-negative heltal, hvoraf det første skal være 0, er vi interesserede i at tælle antallet af "omvendinger" i listen. En omvending er en position i listen, hvor elementerne i stedet for at blive større og større begynder at blive mindre, eller omvendt. For listen:

$$(0, 2, 7, 19, 17, 14, 7, 7, 9, 10, 2, 88, 11)$$

vil svaret således være 5, på grund af omvendingerne ved elementerne 17, 9, 2, 88 og 11.

Formelt kan vi for en liste S definere *retningen* af et indeks, $R(j)$, som følger:

$$R(0) = 1$$
$$R(j) = \left\{ \begin{array}{rl} 1 & \text{hvis S.}(j-1) < \text{S.}(j) \\ -1 & \text{hvis S.}(j-1) > \text{S.}(j) \\ R(j-1) & \text{hvis S.}(j-1) = \text{S.}(j) \end{array} \right\} \text{for } j > 0$$

En omvending er så et indeks j for hvilket $0 < j < |\text{S}|$ og $R(j-1) \neq R(j)$.

Følgende algoritme løser dette problem:

Algoritme: Omvendinger
Stimulans: S: **List**(Int), S.(0)=0
Respons: n: Int, n $= |\{j \mid 0 < j < |\text{S}| \ \wedge \ R(j-1) \neq R(j)\}|$
Metode:
```
r, n, i:=1, 0, 1
do { I }
    i< | S |  →
     if r*(S.(i)-S.(i-1))<0 → r, n:=-r, n+1 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%)

En tabel indeholder n rækker hver med m heltal. Vi er interesserede i at afgøre, om der findes to forskellige rækker, der indeholder de *samme* heltal (men ikke nødvendigvis i samme rækkefølge). I eksemplet:

17	212	8
47262	328	999
8	17	212
1117	5	839

er svaret "ja" på grund af den første og den tredje række.

a) Angiv en effektiv algoritme, der løser dette problem. Hvad er algoritmens tidskompleksitet?

Vi ændrer nu udgangspunktet, så tabellen indeholder bogstaver i stedet for heltal.

b) Angiv en algoritme, der løser det samme problem men udnytter, at vi behandler bogstaver i stedet for heltal. Hvad er algoritmens tidskompleksitet?

Opgave 4 (20%)

Der skal konstrueres en box med følgende udseende:

> **Box** SuperStack
>   **Type** S = ≪stak af heltal≫
>   **Proc** Init [s: S]
>   **Proc** Push [s: S] (x: Int)
>   **Proc** Pop [s: S] → (Int)
>   **Proc** Empty [s: S] → (Bool)
>   **Proc** Member [s: S] (x: Int) → (Bool)
>   **Proc** Position [s: S] (x: Int) → (Int)
> **end** SuperStack

Init, Push, Pop og Empty fungerer som for en almindelig stak (jfr. [P&D], afsnit 2.1.1). Member [s] (x) afgør, om x findes i stakken s. Position [s] (x) antager, at x findes i stakken s og angiver dets afstand fra stakbunden (den *mindste* afstand, hvis x findes flere gange i stakken s).

a) Giv en formel specifikation af operationen Position.

b) Angiv en implementation af typen S, så Init [s] og Empty [s] har tidskompleksiteter i $O(1)$ og Push [s] (x), Pop [s], Member [s] (x) og Position [s] (x) alle har tidskompleksiteter i $O(\log |s|)$.

Opgave 5 (20%)

I denne opgave betragter vi relationer over kursustilmeldinger. Relationerne har alle skemaet:

`årskort:text`	`kursus:text`

På sådanne relationer indfører vi en ny operator `top(R)`, defineret som:

```
max(!(R)|kursus:rel(# << tup(antal:|@(1)|)),antal)
```

Resultatet af `top(R)` vil således altid være et heltal. Lad `Daimi` være følgende eksempelrelation:

årskort:text	kursus:text
950001	Dat1
950001	Mat10
950001	Mat11
950002	Dat1
950002	Mat10
950003	Dat1
950004	Dat1
950004	Mat11

a) Angiv `top(Daimi)`.

b) Forklar i ord, hvad `top(R)` beregner.

c) Er regnereglen `top(R+S)` = `top(R)+top(S)` gyldig? Begrund dit svar.