Opgave 1 (15%)

I denne opgave skal man skrive en TRINE procedure, der kontrollerer lovligheden af et simpelt relationsudtryk. Oplysningerne om et sådant udtryk præsenteres som en værdi af følgende type

Type RelExp = **Sum**(rel: Name'Set, join,union: Pair, project: Proj)
Type Proj = **Prod**(p: RelExp, n: Name'Set)
Type Pair = **Prod**(l, r: RelExp)

Bladene svarer til konstante relationer, om hvilke man kender mængden af attributnavne. De indre knuder svarer til relationsoperationer; join og union tager hver to relationsargumenter (angivet ved l og r); project tager et relationsargument (angivet ved p) og en mængde af attributnavne, over hvilke der skal projiceres (angivet ved n).

Boxen Name, der kan antages at være til rådighed, implementerer en datatype, hvis værdier er mængder af navne. Den har blandt andre operationerne Union og Subset, der er specificeret som følger

```
Box Name
    Type Set = ≪mængde af navne≫

    ⋮

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

    Proc Subset(s1,s2: Set) → (Bool)
        return ≪s1⊆s2≫
    end Subset
end Name
```

Et relationsudtryk er *lovligt*, hvis de to relationsargumenter til union har samme mængde af attributnavne, og hvis relationsargumentet til project indeholder alle de attributnavne, der skal projiceres over.

Resultatet af en lovlig union har samme attributnavne som argumenterne; resultatet af en join har som attributnavne foreningsmængden af argumenternes attributnavne; og resultatet af en lovlig project har som attributnavne den mængde, der projiceres over.

Skriv en TRINE værdiprocedure

Proc FindAtts[R: RelExp] → (Name'Set)

Hvis udtrykket er ulovligt returneres standardværdien; ellers returneres mængden af attributnavne i resultatet af relationsudtrykket beskrevet af R. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Den *transponerede* M^t af en kvadratisk $n \times n$ matrix M opnås ved at spejle i hoveddiagonalen. Fx gælder der, at

$$(1)^t = (1)\,, \quad \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^t = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}^t = \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

Der skal skrives en procedure

```
Proc Transpose[M: List(List(Int))]
```

der transponerer M under forudsætning af, at denne er kvadratisk.

> **a)** Angiv et prædikat Square(M), som udtrykker, at M er kvadratisk. Angiv dernæst en formel specifikation af Transpose.

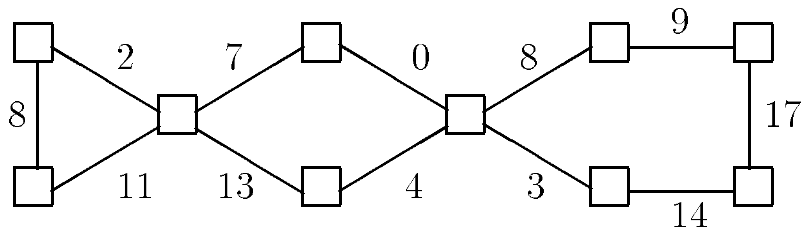
Proceduren kan skrives på følgende måde.

```
Proc Transpose[M: List(List(Int))]
    (+ Var i: Int
       Var H: List(List(Int))
       i, H := |M|,M
       do { ∀j, k ∈ i..|M| : H.(j, k) = M.(k, j) }
           0<i →
               ≪opdater H≫
               i := i-1
       od
       M := H
    +)
end Transpose
```

> **b)** Udfyld ≪opdater H≫, så proceduren bliver korrekt. Bevis, at den opfylder sin specifikation.

Opgave 3 (15%)

En ***cykelkæde*** er en vægtet graf, der består af en kæde af simple cykler, kaldet ***led***. To naboled har altid en enkelt knude til fælles, kaldet en ***stift***. I en cykelkæde har stifterne således grad 4, medens de øvrige knuder har grad 2. Det følgende er et eksempel på en cykelkæde med tre led og to stifter.



Antag, at en cykelkæde har k led, n knuder og m kanter.

a) Angiv m udtrykt ved k og n.

b) Beskriv i ord, hvordan man kan finde det letteste udspændende træ for en cykelkæde i tid $O(n)$.

Opgave 4 (20%)

Der skal konstrueres en box Pitcher med følgende udseende

> **Box** Pitcher
>   **Type** P = ≪sæk af heltal≫
>   **Proc** Init[p: P]
>   **Proc** Insert[p: P](i: Int)
>   **Proc** Remove[p: P](i: Int)
>   **Proc** Freq[p: P] → (**Prod**(f: Int, l: **List**(Int)))
> **end** Pitcher

som realiserer en datastruktur, hvis værdier er ***sække*** af heltal.

Proceduren Init giver den tomme sæk. Proceduren Insert indsætter en forekomst af tallet i. Proceduren Remove fjerner en forekomst af tallet i. Proceduren Freq, der antager at sækken ikke er tom, giver den største frekvens af et tal i sækken, samt listen af de tal, der optræder med denne frekvens. For sækken

$$< 0, 3, 4, 4, 7, 8, 8, 8, 9, 11, 11, 11, 13, 17, 17 >$$

vil Freq således returnere parret $(3, (8, 11))$.

I det følgende angiver $||\mathrm{p}||$ antallet af ***forskellige*** elementer i sækken p.

Beskriv en realisation af typen P, så Init får tidskompleksitet $O(1)$, Insert og Remove får tidskompleksitet $O(\log ||\mathrm{p}||)$, og Freq får tidskompleksitet $O(k + \log ||\mathrm{p}||)$, hvor k er længden af den liste der returneres. Besvarelsen skal *ikke* indeholde et TRINE program.

Opgave 5 (15%)

En ***ordnet stak*** er en stak i hvilken elementerne forekommer i stigende orden

$\longrightarrow$	7
	5
	3
	0
	-5
	-5
	-6

Den understøtter følgende operationer.

Push: push'er et nyt element på toppen og retablerer ordningsinvarianten ved om nødvendigt gentagent at smide det umiddelbart underliggende element væk indtil elementerne igen forekommer i stigende orden.
Destroy: skriver eventuelle elementer ud og tømmer samtidigt stakken.
Top: returnerer en kopi af stakkens øverste element.

Hvis fx tallet 2 blev **Push**'et på ovennævnte stak ville resultatet blive

$\longrightarrow$	2
	0
	-5
	-5
	-6

idet vi for at retablere invarianten først må fjerne 7, derefter fjerne 5, og endeligt fjerne 3.

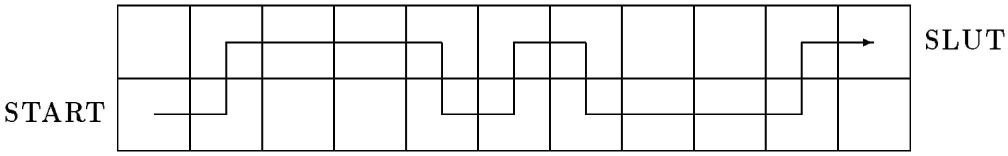
> Angiv en implementation af en ordnet stak, så alle operationer får amortiseret udførelsestid i $O(1)$. Der ønskes en eksplicit angivelse af en nyttig potentialfunktion.
>
> Besvarelsen skal ***ikke*** indeholde et TRINE program.

Opgave 6 (15%)

Vi er givet en $2 \times n$ tabel af ikke-negative heltal som fx

	5	1	7	6	1	2	1	0	6	3	9	SLUT
START	3	0	4	2	5	3	3	6	7	5	6	

To felter er ***naboer*** hvis de står ved siden af hinanden eller over hinanden. En ***nabovej*** er en sekvens af nabofelter fra START til SLUT. For ovenstående tabel er det følgende et eksempel på en nabovej.



Omkostningen af en nabovej er summen af tallene i de felter, den indeholder. Omkostningen af ovenstående nabovej er 62.

> Angiv, hvorledes man med dynamisk programmering effektivt kan beregne omkostningen af den billigste nabovej. Angiv udførelsestiden.