Opgave 1 (20%)

Et udtrykstræ med de fire regnearter, heltalskonstanter og variabler beskrives af følgende rekursive TRINE-type:

Type Expr = **Sum**(plus, minus, times, div: Args, const: Int, name: Text)
Type Args = **Prod**(left, right: Expr)

Vi er interesserede i at udelukke udtryk, der indeholder en division med ***konstanten*** nul. Følgende udtryk skal således begge udelukkes:

```
x/0
(a+b)*(z-(y/3)/0)
```

hvorimod ingen af disse udtryk skal udelukkes:

```
17/x
14/(10-(5+5))
```

(selv om de måske vil give division med nul, når de udregnes).

> Skriv en TRINE-værdiprocedure:
>
> **Proc** Udeluk[E: Expr] → (Bool)
>
> der afgør, om udtrykket E skal udelukkes. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

I denne opgave skal der konstrueres en algoritme, der finder længden af den længste sekvens af 0'er i en vektor L.

Betragt følgende funktioner:

$$ln(L,i,j) = \begin{cases} j-i & \textit{hvis } i \le j \text{ og } L(i..j) = \text{Vector}(0 \mid j-i) \\ 0 & \textit{ellers} \end{cases}$$

$$lhn(L) = \begin{cases} 0 & \textit{hvis } |L| = 0 \\ \underset{0 \le i < |L|}{\textit{max}} \{ln(L, i, |L|)\} & \textit{ellers} \end{cases}$$

a) Forklar i ord, hvad disse funktioner angiver. Skriv dernæst en funktion $lln(L)$, der angiver længden af den længste sekvens af 0'er i L.

Betragt følgende algortime.

Algoritme: Sekvens af 0'er
Stimulans: L: Vector
Respons: m: Int, $m = lln(L)$
Metode: $r := 0$
$\ll$initialiser m og $h \gg$
do $\{ (m = lln(L(0..r))) \wedge (h = lhn(L(0..r))) \wedge (0 \le r \le |L|)\}$
  $r < |L| \rightarrow$
    $\ll$opdater m og $h \gg$
    $r := r+1$
od

b) Konkretiser de ubestemte stumper, så algoritmen bliver gyldig, terminerende og korrekt. Bevis, at dette er tilfældet.

Opgave 3 (20%)

To knuder i en ikke-orienteret graf er ***forbundne***, hvis der findes en vej fra den ene til den anden. Vi skal besvare sådanne spørgsmål om en graf med knuder $1, 2, \ldots n$, i hvilken vi løbende kan tilføje flere kanter.

Der skal konstrueres en box Graf med følgende udseende:

Box Graf
  Type G = ≪ikke-orienteret graf≫
  Proc Init [g: G] (n: Int)
  Proc Insert [g: G] (i,j: Int)
  Proc Connected [g: G] (i,j: Int) → (Bool)
end Graf

Init opretter en graf med n knuder uden kanter, Insert tilføjer kanten {i,j} og Connected afgør, om i og j er forbundne i den aktuelle graf.

a) Angiv hvordan de inverse træer fra datatypen Ækvivalensrelation kan anvendes til at implementere typen G, så **Init**[g](n) har tidskompleksitet i O(n), **Insert**[g](i,j) har tidskompleksitet i $O(\log n)$ og **Connected**[g](i,j) har tidskompleksitet i $O(\log n)$.

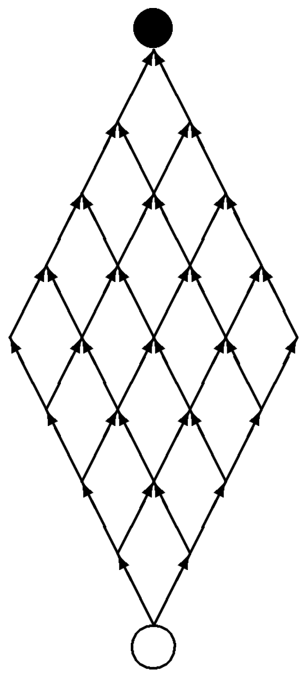
b) Hvordan skal implementationen af G ændres, hvis vi ønsker at tilføje en operation:

Proc EdgeSize [g: G] (i: Int)

som angiver antallet af ***kanter*** i den sammenhængskomponent der indeholder knuden i.

Opgave 4 (25%)

En ***isvaffel*** er en vægtet, orienteret graf, hvor kanterne sidder i et kvadratisk gitter som følger:



Knuden med indgrad 0 kaldes ***spidsen***, og knuden med udgrad 0 kaldes ***kirsebærret***. Vi definerer ***graden*** af en isvaffel til at være antallet af knuder langs en af siderne. En isvaffel af grad k har således k^2 knuder i alt. Tegningen ovenfor er dermed en skitse af en isvaffel af grad 5 (hvor kanternes vægte dog ikke er angivet).

Vi ønsker at finde længder af korteste veje i en isvaffel.

a) Angiv, udtrykt ved graden k, tidskompleksiteten af at finde længden af den korteste vej fra spidsen til kirsebærret ved hjælp af Dijkstras algoritme.

b) Angiv, hvorledes dynamisk programmering også kan bruges til at løse problemet under a). Hvad bliver tidskompleksiteten?

c) Angiv den mest effektive løsning, du kan finde, for at beregne længderne af de korteste veje mellem alle par af knuder. Hvad er tidskompleksiteten, igen udtrykt ved k?

Opgave 5 (15%)

RASMUS-relationen K indeholder information om kurser og deres størrelser målt i studiepoints:

kursus:Text	points:Int
Dat 1	4
Mat 10	2
Mat 11	2
⋮	⋮

RASMUS-relationen S indeholder information om studenter og deres eksamensforsøg:

årskort:Text	kursus:Text	karakter:Int
930001	Dat 1	8
930001	Mat 10	5
930002	Mat 11	7
930087	Dat1	13
⋮	⋮	⋮

a) Er det følgende et lovligt RASMUS-udtryk? Begrund dit svar.

```
!(S*K): rel(# << tup(beståaet: #.karakter>5))
```

b) Beregner følgende to RASMUS-udtryk samme værdi? Begrund dit svar.

```
S|+kursus,karakter
((S|+årskort,kursus)*(S|+årskort,karakter))|-årskort
```

Det vægtede karaktergennemsnit for en student er defineret som summen over alle beståede kurser af karakteren ganget med kursets studiepoints, slutteligt divideret med summen af de beståede studiepoints.

c) Skriv et RASMUS-udtryk, der for en student med årskort `x` beregner det vægtede karaktergennemsnit.