Opgave 1

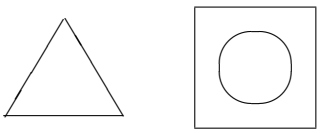
En *figurtegning* kan beskrives på følgende måde

- **trekant**, **firkant** og **cirkel** beskriver tegninger af de tilsvarende simple figurer.
- T_1 **til venstre for** T_2 beskriver en tegning af T_1 ved siden af T_2, centreret vandret.
- T_1 **oven på** T_2 beskriver en tegning af T_1 oven på T_2, centreret lodret.
- T **inde i** S beskriver en tegning af T inde i den simple figur S.

For eksempel ser

trekant til venstre for (cirkel inde i firkant)

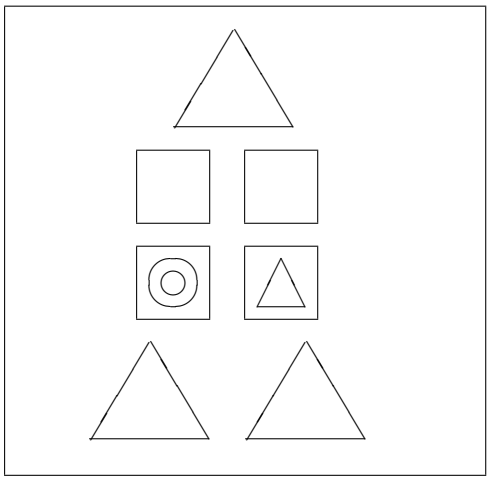
ud som følger



a) Angiv tegningen beskrevet af

((((firkant inde i firkant) oven på cirkel) til venstre for ((cirkel oven på trekant) inde i trekant)) til venstre for (firkant inde i cirkel)) oven på (trekant inde i trekant)

b) Angiv beskrivelsen af følgende tegning



Er der mere end én beskrivelse, der passer?

c) Hvorfor er det nødvendigt med parenteser i sådanne beskrivelser?

Opgave 2

Det følgende er eksempler på romertal og deres decimale værdier

III	=	3
XIX	=	19
LVII	=	57
D	=	500
MCMXCII	=	1992

Derimod er nedenstående bogstavfølger *ikke* romertal

ABC
IIII
MLL
XCX
CCM

a) Giv en præcis beskrivelse af, hvilke bogstavfølger der er romertal.

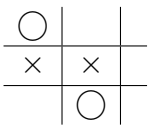
b) Beskriv, hvordan man udregner den decimale værdi af et romertal.

c) Byt svar på a) og b) med en anden på dit øvelseshold. Er svarene præcise nok til at kunne bruges af andre? Er de korrekte?

d) De "rigtige" romertal har den egenskab, at enhver talværdi kan udtrykkes på netop én måde. Kunne man med fordel definere nogle "generaliserede" romertal, der ville gøre det lettere at løse a) og b)?

Opgave 3

Betragt et almindeligt 3×3 kryds-og-bolle spil, hvor de to spillere skiftes til at sætte $\times$'er og $\bigcirc$'er. Man vinder spillet ved at sætte 3 af sine egne symboler på stribe – lodret, vandret eller diagonalt. Hvis brættet bliver fyldt op inden dette sker, så ender spillet uafgjort. En *tilstand* i spillet er et billede af brættet som fx



a) Vis, hvilke tilstande ovenstående tilstand kan udvikle sig til i næste træk. Antag, at $\times$ altid begynder et spil.

b) Det påstås, at hvis en spiller aldrig dummer sig, så vil et spil altid ende uafgjort. Hvordan kan man (i princippet) afgøre dette?

Opgave 4

I et spil med to deltagere starter man med tallene fra 1 til 9 på bordet. Hver spiller skiftes til at tage et tal. Vinderen er den, der først har tre tal, hvis sum er 15.

a) Hvis ingen dummer sig, vil spillet så altid ende uafgjort?

b) Hvad sker der, hvis vi starter med også tallet 10 på bordet?

Opgave 5

Betragt som eksempel på næsten normal prosa-beskrivelse af en indviklet proces vedlagte uddrag af forslag til valgregulativ for Foreningen af Danske Lægestuderende.

```
§8  stk. 1  Valget foregår som forholds-prioritetsvalg.
    stk. 2  Efter sorteringen forsynes stemmesedlerne med en
            fortløbende nummerering, der skal anvendes i den
            talmæssige behandling af stemmeopgørelsen.
    stk. 3  Valget foregår ved en række bortvalg, som
            hvert resulterer i udelukkelse af netop en kandidat
            og foreløbig valg af de øvrige tilbageværende
            kandidater.
    stk. 4  Efter hvert bortvalg stryges den udelukkede kandidat
            fra samtlige stemmesedler.
    stk. 5  Valget er afsluttet, når antallet af tilbage-
            værende kandidater netop er antallet af kandidater,
            der skal vælges.
    stk. 6  Et bortvalg foregår på følgende måde:
            Hver stemmeseddel gives vægten 1, som tildeles den
            kandidat, der er nævnt først på stemmesedlen.
    stk. 7  Summen af de vægte, der er tildelt en kandidat,
            benævnes kandidatens stemmesum.
    stk. 8  En kandidat er foreløbig valgt, hvis hans stemme-
            sum er større end den gennemsnitlige stemmesum for
            de tilbageværende kandidater.
    stk. 9  Forskellen mellem en foreløbig valgt kandidats
            stemmesum og gennemsnittet fordeles som ny vægt på
            de stemmesedler, der har valgt kandidaten, i
            forhold til deres gamle vægt.
    stk. 10 Processen gentages, indtil alle kandidater på
            nær en er foreløbig valgt, idet vægtene nu
            tildeles den først nævnte kandidat, der endnu ikke er
```

```
          foreløbigt valgt.
stk. 11 Såfremt alle kandidaters stemmesum ved den i
          stk. 7 nævnte proces netop er gennemsnittet, foretages
          bortvalget ved lodtrækning mellem disse kandidater.
```

Det følgende er stemmesedlerne i et minivalg, hvor der er fire kandidater A, B, C og D. Der skal vælges én kandidat.

1	2	3	4	5	6	7	8	9	10
A	A	A	A	B	B	B	C	C	D
C	C	C	C	A	C	D	D	D	B
D	D	D	D	D	D	A	A	B	C

a) Anvend §8 på ovenstående eksempel.

b) Er beskrivelsen i §8 fuldstændig og utvetydig?

Opgave 6

Diskuter følgende algoritmiske problemer, og angiv om muligt løsninger. Hvilke primitiver benytter du?

a) Givet to tal x og y i den sædvanlige decimale repræsentation (som fx $x = 37933$ og $y = 5316455617$) skal vi finde deres produkt $z = x \times y$.

b) Givet en dansk tekst skal vi afgøre, hvor svær den er at læse.

c) Givet en dansk tekst skal vi beregne dens LIX-tal, som er (gennemsnitligt antal ord pr. meningsenhed) + (den % af ordene, der indeholder mindst syv tegn).

d) Givet et antal byer skal vi planlægge den bedste rute, efter hvilken man kan besøge dem alle.

Opgave 7

Denne opgave omhandler grammatikker.

a) Hvilke tegnfølger defineres af **Pal** i følgende grammatik

Pal ::= `0` **Pal**$^\circ$ `0` |
`1` **Pal**$^\circ$ `1` |
`0` | `1`

b) Skriv en grammatik, der definerer mængden af tegnfølger bestående af `a`'er og `b`'er, i hvilke der er lige mange `a`'er og `b`'er.

c) En detektiv observerer personer, der går ind og ud af et hus med kun én indgang. I rapporten skrives et `I`, hver gang en person går ind, og et `U`, hver gang en person går ud. Fx betyder rapporten

```
IIUU
```

at to personer gik ind og derefter ud igen. En rapport kan derimod ikke se ud som

```
IUUI
```

da huset jo er tomt ved det andet `U`. Skriv en grammatik, der definerer mængden af mulige rapporter fra detektiven.

Opgave 8

Denne opgave ser på grammatikker.

Vis, at man kan undvære operatorerne + og ∗. Det vil sige, vis hvordan en grammatik der bruger + og ∗ kan skrives om til en grammatik der ikke bruger dem, men som definerer de samme tegnfølger.

Opgave 9

Betragt følgende sætninger. Angiv tilstandstabellerne ved mærkerne {*}.

a)
```
(+ Var x, y, z: Int
   {*}
   x := 87
   y := 105
   {*}
   x, y, z := y, x, y
   {*}
+)
```

b)
```
(+ Var a, b, c: Int
   (+ Var a, b: Int
      {*}
      a, b, c := 11, 22, 33
   +)
   {*}
+)
```

c)
```
(+ Var r, s, t: Int
   if true → r, s, t := 87, 88, 89
   | true → r := 22
```

```
    | true → r:=11
    fi
    {*}
    if r>20 → s, t:=101, 102 fi
    {*}
+)
```

Opgave 10

Skriv og kør programmer, der

a) indlæser et bogstav og ændrer det fra stort til lille, eller fra lille til stort.

b) indlæser en række tal, afsluttet med 0, og udskriver det største og det mindste tal.

c) indlæser 5 tal og udskriver dem i sorteret orden (ikke-aftagende).

Opgave 11

Betragt veksleprogrammet på side 20 i TRINE noten.

a) Angiv samtlige værdiudtryk, variabeludtryk og sætninger. Angiv for hvert af udtrykkene dets type, og for hver sætning dens "slags", som angivet i kapitel 2.

b) Bekræft, at programmet er lovligt ifølge grammatikken i appendix A i TRINE noten.

c) Skriv en deterministisk version af programmet, der altid udbetaler så *få* mønter som muligt.

d) Kan du løse c), hvis vi også indfører en 7-krone mønt?

Opgave 12

Følgende sætning vil indlæse k heltal og udskrive dem i sorteret orden

```
(+ Var x1, x2, ... xk: Int
   read[ x1, x2, ... xk]
   do x1>x2 → x1:=:x2
    & x2>x3 → x2:=:x3
     ⋮
    & xk-1>xk → xk-1:=:xk
   od
   write(x1, " ", x2, " ",...," ", xk)
+)
```

Dette virker dog kun for et fast k.

Skriv en sætning, der indlæser en liste af heltal og udskriver dem i sorteret orden. Sætningen skal så nøje som muligt efterligne ovenstående **do**-sætning.

Opgave 13

Et simpelt undervisningsprogram kan udformes som en dialog mellem programmet og eleven. Det kan fx se således ud (eleven skriver de linjer der er understreget)

> Hvad er 25+47?
> 62
> Forkert. Prøv igen.
> 72
> Rigtigt.
> Hvad er 435+47?
> 473
> Forkert. Prøv igen.
> 472
> Stadig forkert. Svaret er 482.
> Hvad er 33+102?
> ⋮

Der skal skrives et program, der realiserer sådanne dialoger.

a) Angiv den overordnede struktur af programmet.

b) Skriv og kør et program, der løser problemet og følger strukturen angivet i a). Bemærk, at Int-udtrykket random(i, j) angiver en tilfældig af værdierne i, i+1, ..., j-1.

c) Udvid eventuelt programmet, så det tilbyder mere avancerede regnestykker, fører statistik, og måske tilpasser sværhedsgraden til elevens niveau.

Opgave 14

I det følgende er $N = \{0, 1, 2, 3, \ldots\}$, $B = \{\text{true}, \text{false}\}$ og $T = \emptyset$. Angiv mængderne

a) N^*

b) $B \times N$

c) $N + N$

d) $B + N$

e) T^*

f) $T \times N$

g) $T + B$

h) $((N \times B) + B)^*$

Opgave 15

Betragt følgende program

```
Process P
    (+ Var a, b: Int
       Var c: Vector
       a, c := 87, Vector(11, 22, 33)
       (*1*)
       (+ Var b, d: Int
          Var e: Vector
          e, c, b := c, Vector(), c.(0)
          (*2*)
       +)
    +)
end P
```

Angiv programmets tilstandstabel ved `(*1*)` og `(*2*)`.

Opgave 16

Et spillekort kan defineres med typen

Type Kort = **Sum**(hjerter, ruder, klør, spar: Int)

således at hjerter 5 angives af udtrykket

Kort(hjerter: 5)

En samling af kort kan defineres med typen

Type Spil = **List**(Kort)

a) Skriv en sætning, der sætter en variabel S af type Spil til at indeholde et normalt spil kort (52 stykker).

b) Skriv en sætning, der udskriver værdien af en variabel S af type spil på en pæn måde.

c) Skriv en sætning, der blander kortene i en variabel S af type Spil. Vink: Man kan "tage af" med følgende sætning

```
S := S(i .. |S|) ++ S(0 .. i)
```

når værdien af i ligger mellem 0 og længden af S.

d) Man kan også benytte typerne

```
Type Farve = Sum(hjerter, ruder, kløv, spar:Unit)
Type Kort = Prod(f: Farve,v: Int)
Type Spil = List(Kort)
```

Bliver der nogen forskel?

Opgave 17

Nedskriv typer, hvis værdier behændigt kan repræsentere nedenstående informationer. Angiv et konstant værdiudtryk af hver af de nedskrevne typer.

a) breve

b) dProg1-studenter

c) en situation i kryds-og-bolle

d) en bankkonto

Opgave 18

Betragt de to programmer i noten side 34 og 35, der sætter en vektor til at indeholde værdien $(0, 1, 2, \ldots, n-1)$. Kør begge programmer med forskellige n-værdier og mål kørselstiderne (det kan man gøre med Int-udtrykket clock der angiver systemets tid i millisekunder). Lav to kurver på millimeterpapir, der viser tiderne som funktioner af n. Forklar resultaterne.

Opgave 19

Lad v angive en persons legemsvægt i kg og a det antal gram alkohol vedkommende har i blodet. Hvis man i løbet af en time indtager g genstande, så har a efter denne times forløb ændret sig som følger

$$\begin{array}{ll} a := (a - f * v) + g * 12 & \textbf{hvis } a > f * v \\ a := g * 12 & \textbf{hvis } a \leq f * v \end{array}$$

hvor $f = 0.1$ angiver forbrændingen i gram/time/kg. Alkoholpromillen kan nu findes som

$$p = a/(v * k)$$

hvor $k = 0.68$ for mænd og $k = 0.55$ for kvinder (da der er forskel på mænds og kvinders fedtvæv).

a) Skriv et program, der indlæser køn og vægt samt en liste med hvor mange genstande, der indtages i hver af en række timer, og udskriver en ligeså lang liste med personens promille i de samme timer.

b) Modificer programmet, så det udskriver hele "henfaldet af beruselsen", det vil sige, time for time angiver promillen indtil den bliver 0.

Virker formlen rimelig?

Opgave 20

Betragt reciprokprogrammet på side 15 i TRINE noten. En reciprokværdi $1/n$ kan skrives på formen

$$0.\alpha\beta\beta\beta\beta\ldots$$

hvor α og β er valgt entydigt kortest. For $1/6$ har vi således $\alpha = 1$ og $\beta = 6$, for $1/7$ er α tom og $\beta = 142857$, og for $1/50$ er $\alpha = 02$ og $\beta = 0$. Skriv en sætning, der indlæser n og udskriver α og β.

Opgave 21

RASMUS relationen `folkeskole` kan repræsenteres som en værdi af typen Relation

Type Tupel = **Prod**(id: Int, afgang: Text, karakter: Int, egnet: Text)
Type Relation = **List**(Tupel)

På filen `eksempler/folkeskole` ligger den tilsvarende relation i et format, der kan load'es ind i en variabel af type Relation. I denne opgave skal der laves forskellige database manipulationer på denne relation.

a) Indlæs relationen i en variabel F og udskriv antallet af elever, der har afgangseksamen fra niende klasse.

b) Lav en ny relation N, der indeholder den delrelation af F, der omfatter disse elever. Dette svarer til RASMUS udtrykket

```
F ? #.afgang="NI"
```

c) Konstruer og udskriv et *histogram* over karaktererne i N, det vil sige, et antal linjer af formen

```
⋮
87: ***
88: **
89: *****
⋮
```

hvor der ud for hvert tal er én stjerne for hver elev med netop denne karakter. Der bør ikke udskrives linjer uden stjerner.

d) Skriv eventuelt en alternativ version af programmet, der udskriver histogrammet på den anden led, som fx

```
          *
          *
      *   *
      *  *  *
      *  *  *
... 87 88 89 ...
```

Opgave 22

Betragt studenterrelationen på side 53 i TRINE noten. Skriv sætninger, der

a) fjerner alle, der ikke har bestået.

b) deler den op i tre delrelationer A, S og K, der indeholder henholdsvis afmeldte, syge og resten.

c) udskriver alle data for studenter med årskort fra '96.

En passende relation kan load'es fra filen `eksempler/årgang96`.

Opgave 23

a) Angiv (mindst) to forskellige sætninger, der leder frem til tilstanden på side 48 i TRINE noten.

b) Angiv mulige typer for følgende anonyme konstanter

Sum(7: 7)
List(**List**(), "abc")
Prod(**Prod**(**Prod**(1), 2), 3)
Prod(**List**(), 124, **Prod**(true, 'a'))
List(**Sum**(2: 87), **Sum**(3: true), **Sum**(1: "abc"))

c) Beregn normalformen af typen A under følgende definitioner

Type A = **Prod**(x: B, y: Int, z: **Sum**(a: B, b: C))
Type B = **List**(C)
Type C = **Prod**(r: **List**(D), s: E)
Type D = **List**(E)
Type E = **Prod**(p: Int, q: Bool)

Nedskriv et konstant værdiudtryk af hver af disse typer.

Opgave 24

Lad N betegne mængden af naturlige tal, og lad H betegne mængden af alle mennesker. Angiv relationer over N og H der er

a) refleksive, symmetriske, ikke transitive

b) ikke refleksive, ikke symmetriske, transitive

c) ikke refleksive, symmetriske, ikke transitive

d) ækvivalensrelationer

Et (artigt) svar på c) kunne være "... har ingen fælles primfaktorer med ..." og "... har været i biografen med ...", idet næppe alle mennesker i verden har været i biografen.

Opgave 25

Betragt følgende typer

Type A = **Prod**(x: Int, y: C)
Type B = D
Type C = **List**(F)
Type D = **Prod**(x: Int, y: E)
Type E = C
Type F = **Sum**(a: Int, b: Real)

Vis, at typerne A og B er ækvivalente

a) ved at argumentere ud fra reglerne.

b) ved at beregne normalformerne.

Opgave 26

En samling af spillekort kan som beskrives ved typerne

Type Kort = **Sum**(spar, hjerter, ruder, klør: Int)
Type Spil = **List**(Kort)

a) Skriv en sætning, der tager en variabel hånd af type Spil og skriver den ud som en "korthånd", fx

S K Q 10 8
H A K 7
R Q J 9 2
K 8 3

hvor kortene altså skal være sorterede efter *farve* og *værdi*. Sætningen skal laves som en trinvis forfinelse af strukturen

```
(+ Var sorteret: Bool
   sorteret := false
   do ¬ sorteret →
         (+ Var i: Int
            sorteret := true
            i := 1
            do i<| hånd | →
                  ≪ opdater kort og sorteret ≫
                  i := i+1
            od
         +)
   od
   ≪ udskriv den sorterede hånd ≫
+)
```

b) Skriv et program, der blander et spil kort (se opgave **16**), udtager en hånd på tretten kort og skriver den ud.

c) Diskuter hvordan man kunne skrive et program, der giver kort til en omgang bridge (4×13 kort) og skriver samtlige hænder ud.

Opgave 27

Pe-pe-sproget er en variant af dansk, hvis brug hjælper med til at gøre selv den simpleste talestrøm ganske uforståelig…

Man laver en almindelig sætning om til pe-pe-sprog ved at dele alle ord op i stavelser. I pe-pe-sproget gentager man alle stavelserne to gange, med den modifikation, at anden gang stavelsen siges, udskiftes de indledende konsonanter med et 'p'. Følgende er nogle simple eksempler:

"Trine"	→	"Tripinepe"
"Rasmus"	→	"Raspasmuspus"
"Datalogi"	→	"Dapatapalopogipi"
"Ugeseddel"	→	"Upugepesedpeddelpel"

a) Konstruer et program, der indlæser ord fra skærmen og udskriver de tilsvarende pepeord. Man kan lade sig inspirere af orddelings-eksemplet fra TRINE noten side 68. Programmet skal udvikles med "overdreven" brug af trinvis forfinelse.

b) Skriv eventuelt et program, der oversætter den anden vej.

Opgave 28

Et *induktionsbevis* af udsagnet $\forall n \geq 0 : P(n)$ har følgende udseende:

1) Et bevis for $P(0)$.

2) Et bevis for, at hvis man antager $P(n)$ så gælder $P(n+1)$.

Det er et grundlæggende aksiom i matematikken, at denne teknik virker. Benyt induktion til at vise nedenstående udsagn; bemærk, at man først skal formulere dem, så de passer i ovenstående skema.

a) En mængde med n elementer har altid 2^n forskellige delmængder

b)
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

c)
$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Opgave 29

Vis, at sorteringsprogrammet i opgave **12** terminerer.

Opgave 30

a) Vis, at følgende algoritme er gyldig og korrekt.

$\{ (n \geq 1) \wedge (p \geq 0) \}$
r, q := 1, p
do $\{ r * n^q = n^p \wedge q \geq 0 \}$
  $q \neq 0 \rightarrow$ r, q := r*n, q-1
od
$\{ r = n^p \}$

b) Find en passende invariant og vis, at følgende algoritme er gyldig og korrekt.

```
V, i:=Vector(0 | n), 0
do { I }
    i<n → V.(i), i:=i, i+1
od
{ V=Vector(0, 1, 2, ..., n-1) }
```

Opgave 31

Vi betragter en Real-vektor V, i hvilken alle elementer er ≥ 0. Det *maksimale delprodukt* af V er defineret ved

$$\text{maxprod}(V) = \max\{V.(j)V.(j{+}1)V.(j{+}2)\cdots V.(i{-}1) \mid 0 \leq j \leq i \leq |V|\}$$

Vi har fx, at for

$$\begin{aligned} V &= (5.7,\ 2.3,\ 0.4,\ 8.2,\ 2.7,\ 0.999) \\ W &= (0.5,\ 0.2,\ 0.87) \end{aligned}$$

så er $\text{maxprod}(V) = V.(0)\cdots V.(4) = 116.10216$ og $\text{maxprod}(W) = 1$ (bemærk, at et "tomt" produkt har værdi 1). Det følgende er en skitse af en algoritme, der beregner det maksimale delprodukt.

Algoritme: Maksimalt Delprodukt
Stimulans: V: **List**(Real), $\forall i \in 0..|V| : V.(i) \geq 0$
Respons: $m = \text{maxprod}(V)$
Metode: $m, p, i := 1, 1, 0$
do $\{I\}$
$i < |V| \rightarrow$
≪iterer≫
$i := i + 1$
od

Invarianten I siger, at

$$\begin{aligned} m &= \text{maxprod}(V(0..i)) \\ p &= \max\{V.(j)V.(j{+}1)V.(j{+}2)\cdots V.(i{-}1) \mid 0 \leq j \leq i\} \end{aligned}$$

Udfyld ≪iterer≫, så algoritmen bliver gyldig og korrekt. Bevis, at algoritmen er korrekt.

Opgave 32

Bevis at følgende program er gyldigt og terminerer, og at det for et givet n > 0 således korrekt udskriver det n'te Fibonaccital F_n.

```
(+ Var n: Int
   read[n]
```

```
    if n < 1 → abort fi
    (+ Var f, g, k: Int
       f, g, k := 1, 1, 1
       do { (f = F_k) ∧ (g = F_{k−1}) ∧ (1 ≤ k ≤ n) }
            k ≠ n →
                 f, g := f+g, f
                 k := k+1
       od
       { f = F_k }
       write(f)
    +)
+)
```

Opgave 33

Lad i det følgende R og K betegne 2×2 matricer af formen

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \quad K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$

Produktet af R og K er defineret som

$$RK = \begin{bmatrix} r_{11}k_{11} + r_{12}k_{21} & r_{11}k_{12} + r_{12}k_{22} \\ r_{21}k_{11} + r_{22}k_{21} & r_{21}k_{12} + r_{22}k_{22} \end{bmatrix}$$

Vis, at for ethvert $n \geq 1$, så vil beregningen

$K := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$
$R := K^{n-1}$
$f := r_{11} + r_{12}$

sætte f til det n'te Fibonaccital. Hvordan ville du implementere denne algoritme?

Opgave 34

Betragt følgende spil: Man har en krukke med sorte og hvide kugler, dobbelt så mange hvide som sorte. To tilfældige kugler tages op. Hvis de er ens, bliver de smidt væk, og man lægger en ny sort kugle ned i krukken. Hvis de er forskellige, smider man den sorte væk og putter den hvide tilbage.

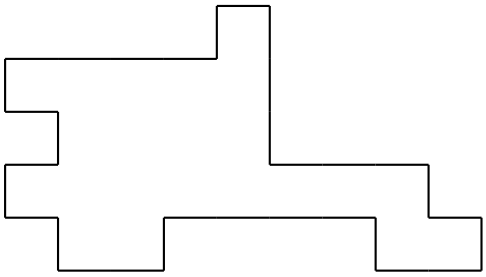
a) Skriv et program, der indlæser antallet af sorte kugler og derefter *simulerer* ovennævnte spil, indtil der kun er én kugle tilbage i krukken.

b) Det er klart at spillet – og dermed programmet – terminerer (hvorfor?). Kan der også siges noget om farven på den sidste kugle?

Opgave 35

To spillere, A og B, skiftes til at sætte hhv. en rød og en blå streg af længde 1 på et stykke kvadreret papir. Hver streg må placeres vilkårligt, vandret eller lodret, men selvfølgelig ikke hvor der er en streg i forvejen.

A begynder (og tegner med rødt), og hans opgave er at lave en sammenhængende lukket "kurve", som fx



der udelukkende består af røde streger. B's opgave er at forhindre A i at lave en sådan kurve.

a) Lad os antage, at du bliver bedt om at lave et program, der kan spille for B. Programmet indlæser A's træk (i en passende repræsentation), og svarer efter hvert træk med B's efterfølgende træk. Er det muligt at skrive dette program, sådan at A aldrig formår at danne en lukket kurve? Hvordan?

   Vink: Opstil en passende invariant for spilleplanens udseende efter hvert B-træk.

b) *Efter at du har lavet* a): Lav en type, der passende kan repræsentere de enkelte træk og skriv programmet.

Opgave 36

Betragt følgende dekorerede sætninger

a)
```
(+ Var i, j: Int
   xio ?? i, j
   if i>j → i:=:j fi
   { i≤j }
   do { i≤j+1 }
        i<j → i:=i+1
      | i<j → j:=j-2
      | i>j+1 → i:=:j
   od
   { i=j }
+)
```

b)
```
(+ Var i: Text
   xio ?? i
   do i>5 → i:=i-1 { i ≥ 5 }
     | i<5 → i:=i+1 { i ≤ 5 }
     | i=5 → i:=i   { i = 5 }
   od
   { i = 4 }
+)
```

- Med hvilke tilstandstabeller er de enkelte udsagn opfyldt?
- Er sætningerne gyldige?
- Hvad laver sætningerne?

Opgave 37

Heltalskvadratroden af et ikke-negativt tal n er det heltal r, der opfylder

$$r^2 \leq n < (r+1)^2$$

a) Argumentér for, at følgende dekorerede udgave af kvadratrodsprogrammet er gyldig, og at programmet beregner heltalskvadratroden.

```
(+ Var n: Int
   read[n]
   if n<0 → abort fi
   (+ Var a, b: Int
      a, b:=0, n
      do { a² ≤ n < (b+1)² }
           n<b*b → b:=b-1
         | a*(a+2)<n → a:=a+1
      od
      write((a+b)/2)
   +)
+)
```

b) Ovenstående program beskriver en *lineær* søgning efter kvadratroden. En bedre måde at finde kvadratroden på, er ved hjælp af såkaldt *binær* søgning, repræsenteret i TRINE ved

```
(+ Var n: Int
   read[n]
   if n<0 → abort fi
```

```
(+ Var a, b, m: Int
   a, b, m := 0, n+1, (n+1)/2
   do { (a² ≤ n < b²) ∧ (m = (a + b)/2) }
        n < m*m →
          b := m
          m := (a+b)/2
      | n ≥ (m+1)*(m+1) →
          a := m+1
          m := (a+b)/2
   od
   write(m)
+)
```

Argumentér som i a) for at programmet er gyldigt og korrekt.

c) Hvorfor er løsningen i b) bedre end den i a)?

Opgave 38

Lad heltalskvadratroden være defineret som i opgave **37**. Bevis at følgende er endnu en korrekt måde at beregne den på.

```
(+ Var n: Int
   read[n]
   if n < 0 → abort fi
   (+ Var q, r: Int
      q, r := 0, 1
      do { r = (q + 1)² ∧ q² ≤ n }
           r ≤ n → q, r := q+1, r+2*q+3
      od
      { q² ≤ n < (q + 1)² }
      write(q)
   +)
+)
```

Opgave 39

Betragt den udvidede Euklids algoritme side 93 i TRINE noten.

a) Vis, at algoritmen også er korrekt såfremt "indmaden" erstattes af

```
p>q →
      (+ Var x: Int
         ≪bestem x så 0<x∗q<p≫
         p, t := p−x∗q, t+x∗s
      +)
| q>p →
      (+ Var x: Int
         ≪bestem x så 0<x∗p<q≫
         q, s := q−x∗p, s+x∗t
      +)
```

b) En vigtig sætning i talteorien siger, at der for vilkårlige positive heltal n og m findes ikke-negative tal a og b, således at $\text{sfd}(n, m) = a * n - b * m$. Skriv en version af Euklids algoritme, der givet n og m beregner a og b. Følgende skitse kan være nyttig.

```
(+ Var n, m: Int
   ≪indlæs og kontroller n og m≫
   (+ Var p, q, a, b, c, d: Int
      ≪initialiser p,q,a,b,c,d≫
      do { (sfd(p, q) = sfd(n, m)) ∧
           (p = a∗n−b∗m) ∧ (q = c∗m−d∗n) ∧
           (p, q ≥ 1) ∧ (a, b, c, d ≥ 0) }
        p>q → ≪reducer p≫
      | q>p → ≪reducer q≫
      od
      ≪udskriv resultatet≫
   +)
+)
```

Opgave 40

Betragt følgende program

```
Proc P[x: Text]
  x.(0) :=: x.(1)
  (∗3∗)
end P

Proc Q[t: Text]
  (+ Var s: Text
     (∗2∗)
     s := t(0 .. 1)
     P[t]
```

```
  +)
end Q

Var t: Text
(*1*)
t:="Trine"
Q[t]
```

Angiv tilstandstabellerne ved `(*1*)`, `(*2*)` og `(*3*)`.

Opgave 41

a) Betragt typen

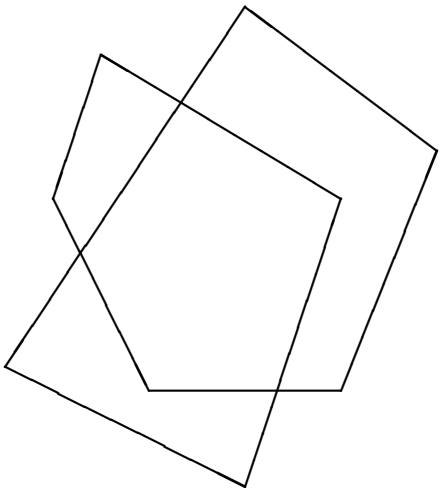
Type Punkt = **Prod**(x, y: Int)

Skriv en værdiprocedure

Proc Afstand(p_1, p_2: Punkt) → (Int)

som beregner afstanden mellem punkterne p_1 og p_2. Afprøv proceduren på passende eksempler.

b) En *polygon* ser ud som følger



Definer en passende type

Type Polygon = ···

og skriv en værdiprocedure, der givet en polygon returnerer dens omkreds.

Opgave 42

Denne opgave drejer sig om beregninger på lange ikke-negative heltal.

a) Vis, hvordan et langt heltal kan repræsenteres som en værdi af typen

Type Lang = **List**(Int)

b) Skriv en procedure

Proc Plus[L_1, L_2: Lang]

der foretager beregningen

$L_1 := L_1 + L_2$

c) Brug proceduren til at beregne Fibonaccital nummer 100 og topotensen 2^{100}.

d) Vis, ved hjælp af udsagn, at proceduren er korrekt.

Opgave 43

LIX-værdien (LæsbarhedsIndeX) for en tekst, der skal opfattes som et mål for tekstens sværhedsgrad, er defineret som: (det gennemsnitlige antal ord pr. meningsenhed) + (den % af ordene, der indeholder mindst syv tegn).

Der skal skrives et program, der beregner LIX-værdien af en indlæst tekst af typen Page = **List**(**List**(Text)). En sådan værdi kan indlæses fra en fil med sætningen

```
loadpage[p]("filnavn")
```

En del af opgaven er præcist at definere, hvad "ord" og "meningsenhed" betyder.

Man kan med fordel starte med at skrive en procedure

Proc Klip[p: Page, i,j: Int] → (Int)

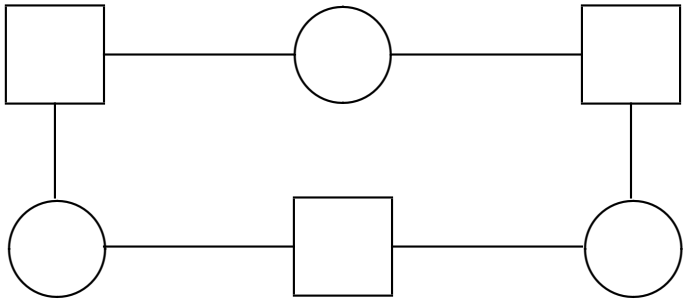
der læser i teksten P fra den i'te linje og det j'te tegn og starter at læse alle ikke-relevante tegn som cifre, blanke og linjeskift. Hvis teksten herfra er tom returneres standardværdien. Hvis teksten indeholder et skilletegn for sætninger returneres 0. Ellers læses den næste sammenhængende sekvens af bogstaver og længden af denne returneres. I alle tilfælde opdateres i og j til at pege efter den nu læste tekst.

Beregner du de samme LIX-værdier som andre på dit øvelseshold?

Opgave 44

Denne opgave omhandler figurbiblioteket beskrevet i Trine noten side 101.

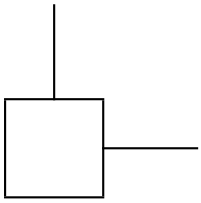
a) Benyt figurbiblioteket til at tegne dette billede



b) Skriv en procedure

Proc Plug(s: Figure)

der tegner følgende billede



hvor alle linjestykker har længde 2∗s.r og kvadratet har centrum i s.ctr.

c) Skriv værdiprocedurer

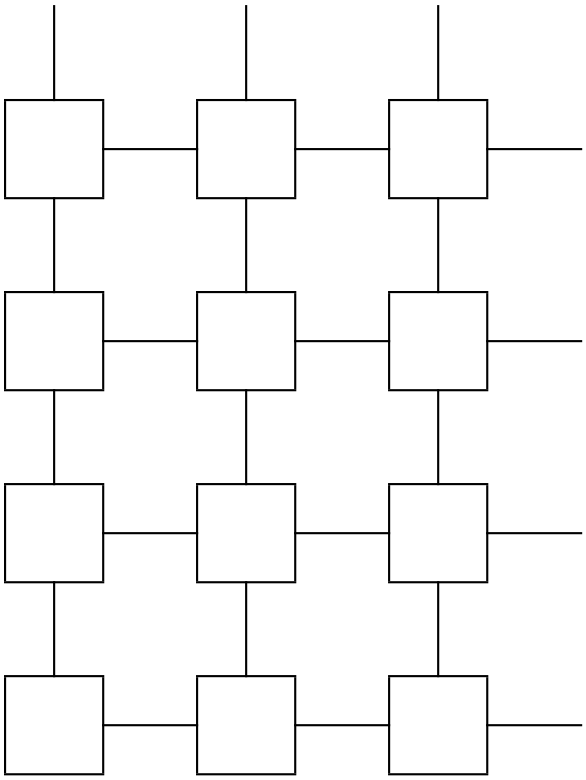
Proc MoveUp(f: Figure) → (Figure)
Proc MoveRight(f: Figure) → (Figure)

der forskyder figuren f afstanden 4∗f.r henholdsvis op og til højre (kun som værdi – der sker intet på skærmen).

d) Skriv en procedure

Proc Grid(n, m: Int)

der tegner følgende n×m billede, sat sammen af individuelle Plugs.



Størrelsen af de enkelte Plugs skal tilpasses, så hele tegningen kan være på skærmen.

Opgave 45

Benyt grafikbiblioteket til at skrive en udgave af alkoholprogrammet fra opgave **19**, hvor henfaldet af beruselsen tegnes som en kurve på skærmen.

Opgave 46

Denne opgave omhandler datatypen Set beskrevet på side 127 i TRINE noten.

a) Udvid datatypen med en operation

```
Proc S'Delete[s: S'Set](i: Int)
    ≪s:=s−{i}≫
end S'Delete
```

b) Brug den udvidede datatype til at løse følgende problem: Indlæs en række tal afsluttet med 0; udskriv de af tallene, der forekommer et ulige antal gange.

Opgave 47

a) Skriv en datatype

```
Box F
    Type Tab = ≪multimængde af tegn≫

    Proc Init[t: Tab]
        ≪t gøres tom≫
    end Init

    Proc Insert[t: Tab](c: Char)
        ≪udvid t med c≫
    end Insert

    Proc Count[t: Tab](c: Char) → (Int)
        ≪giv antal forekomster af c i t≫
    end Count
end F
```

der kan realisere en *multimængde* af tegn (en mængde hvor et element kan forekomme flere gange).

b) Brug F til at lave frekvensanalyse af en tekst, det vil sige en tabel over de indgående bogstaver og deres frekvenser.

Opgave 48

Denne opgave omhandler polyboxen Catalog på side 140 i TRINE noten.

a) Vis, hvordan boxen

```
Box CS
    Catalog(Int, Unit)
end CS
```

og typen CS'Cat kan bruges til at implementere datatypen af heltalsmængder beskrevet på side 128 i TRINE noten (undtagen Print).

b) Vis, hvordan boxen

```
Box CF
    Catalog(Char, Int)
end CF
```

og typen CF'Cat kan bruges til at implementere datatypen af frekvenstabeller beskrevet i opgave **47**.

c) En matrix er *mager*, hvis mange af dens elementer er 0'er. For magre matricer kan vi definere følgende datatype

```
Box M
    Type Mat = ≪en mager heltalsmatrice≫

    Proc Init[M: Mat]
        ≪M sættes til 0-matricen≫
    end Init

    Proc Update[M: Mat](i, j, x: Int)
        ≪M(i, j):=x≫
    end Update

    Proc Inspect[M: Mat](i, j: Int) → (Int)
        ≪returner M(i, j)≫
    end Inspect
end M
```

Brug boxene

```
Box CV
    Catalog(Int, Int)
end CV

Box CM
    Catalog(Int, CV'Cat)
end CM
```

og typen CM'Cat til at implementere boxen M, således at 0'er kun repræsenteres implicit. Vink: Typen Mat skal være CM'Cat.

Opgave 49

a) Nedskriv en type, hvis værdier kan repræsentere telefonbøger. Der skal tages højde for, at telefonnumre kan være hemmelige.

b) Skriv og kør et program, der load'er og dump'er en telefonbog fra/til en fil og håndterer dialoger som den følgende (det, brugeren skriver, er understreget)

opslag
navn: Hans
telefonnummeret til Hans er 86001127

opslag
navn: Birte
Birte er ikke i bogen

optag
nyt navn: Birte
hemmeligt: nej
nyt nummer: 31001482

opslag
navn: Birte
telefonnummeret til Birte er 31001482

opslag
navn: Margrethe
Margrethe har hemmeligt nummer

slut

c) Udvid programmet, så brugeren selv angiver hvilken telefonbog, der skal anvendes.

Opgave 50

Denne opgave drejer sig om at konstruere en datatype, der stiller *rationale* tal til rådighed for programmer. Udgangspunktet er, at ethvert rationalt tal kan skrives som en *normalbrøk*

$$\frac{p}{q} \text{ hvor } p \in \mathbf{Z}, q \in \mathbf{N}$$

og hvor vi yderligere kræver, at når $p = 0$ er $q = 1$, og når $p \neq 0$ er $\text{sfd}(|p|, q) = 1$.

a) Bevis, at ovenstående repræsentation er entydig: hvis p_1/q_1 og p_2/q_2 repræsenterer samme rationale tal på denne måde, så er $p_1 = p_2$ og $q_1 = q_2$.

b) Vis, at følgende sætning beregner største fælles divisor af n og m:

```
do n>m → n:=n mod m
            if n=0 → n:=m fi
  | m>n → m:=m mod n
            if m=0 → m:=n fi
od
```

Hvorfor kunne denne version af Euklids algoritme være at foretrække?

c) Implementer følgende datatype

```
Box Q
    Type Tal = Prod(p, q: Int)

    Proc Konst(p, q: Int) → (Tal)
        ≪returner p/q≫
    end Konst

    Proc Plus(s, t: Tal) → (Tal)
        ≪returner s+t≫
    end Plus

    Proc Minus(s, t: Tal) → (Tal)
        ≪returner s-t≫
    end Minus

    Proc Gange(s, t: Tal) → (Tal)
        ≪returner s*t≫
    end Gange

    Proc Divider(s, t: Tal) → (Tal)
        ≪returner s/t≫
    end Divider

    Proc Større(s, t: Tal) → (Bool)
        ≪returner s<t≫
    end Større

    Proc Læs[t: Tal]
        ≪indlæs tekst af form "8.24" og gem værdien i t≫
    end Læs

    Proc Skriv(t: Tal, n: Int)
        ≪udskriv t med n decimaler≫
    end Skriv
end Q
```

Implementationen skal overholde den invariant, at alle værdier af type Tal er normalbrøker, som beskrevet ovenfor.

d) Kan man sammenligne to Tal ved hjælp af = og <>? Giver det mening? Hvorfor?

Opgave 51

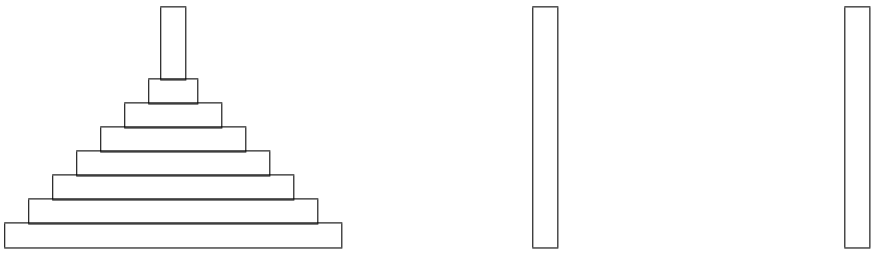
Betragt et binært træ, der fx defineres i TRINE noten på side 152. Vi definerer *højden* af et træ til at være det maksimale antal grene på en vej fra roden til et blad.

a) Hvor mange grene kan der maksimalt være i et binært træ med højde $h \geq 0$?

b) Hvad er den mindst mulige højde af et binært træ med $n \geq 0$ grene?

Opgave 52

Hanoi's Tårne er et spil, der ser ud som følger



Man har 3 pinde og n skiver af forskellige størrelser. Udgangspositionen er som på ovenstående tegning. Opgaven er at flytte alle skiverne fra pind A til pind C, ved hjælp af pind B, idet man aldrig må anbringe en større skive oven på en mindre.

a) Skriv en rekursiv procedure, der givet n udskriver en løsning på formen

```
flyt fra A til C
flyt fra A til B
flyt fra C til B
⋮
```

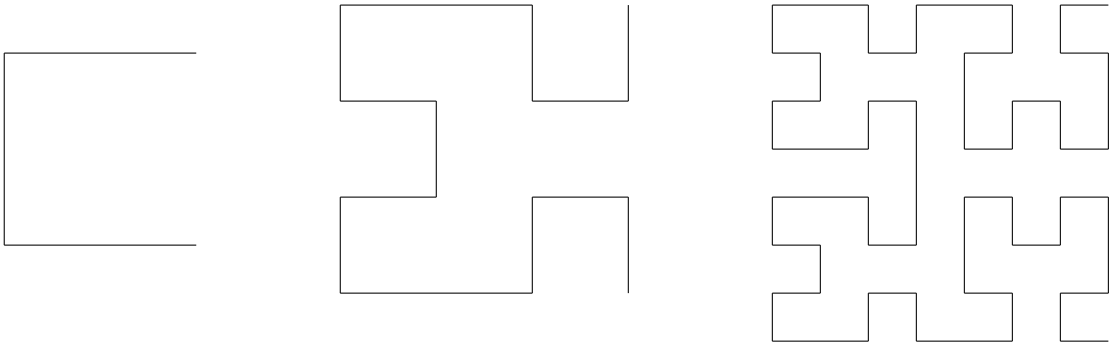
Hvordan ser udførelsestræet ud?

b) Hvor mange flytninger bruger dit program på n skiver?

c) Hvad med 4 pinde, eller flere?

Opgave 53

En *Hilbert kurve* af orden n er den n'te tegning i følgende række, der nummereres H_1, H_2, H_3, og så videre.



Angiv "rekursionsformlen" for H_n. Skriv en rekursiv procedure, der givet n tegner H_n på skærmen. Følgende oplæg kan benyttes

```
Process H
    (+ @"graphics.tri"

        Proc Vest(i, u: Int)
            if i>0 →
                    Syd(i-1, u)
                    turnto(180)
                    move(u)
                    Vest(i-1, u)
                    turnto(270)
                    move(u)
                    Vest(i-1, u)
                    turnto(0)
                    move(u)
                    Nord(i-1, u)
            fi
        end Vest

        Proc Syd(i, u: Int)
            ≪kode for Syd≫
        end Syd

        Proc Øst(i, u: Int)
            ≪kode for Øst≫
        end Øst

        Proc Nord(i, u: Int)
            ≪kode for Nord≫
        end Nord
```

```
        Var n, p: Int

        ≪indlæs n≫
        ≪p:=2^n-1≫
        clear(0)
        jumpto(450, 270)
        Vest(n, 256/p)
    +)
end H
```

Opgave 54

Betragt følgende grammatik, der definerer en lille delmængde af sproget Dansk

Sætning	::=	**Substantiv Bisætning**$^{\bigcirc}$ **Prædikat Fortsættelse**$^{\bigcirc}$
Fortsættelse	::=	, **Konjunktion Sætning**
Bisætning	::=	, som **Substantiv Bisætning**$^{\bigcirc}$ **TransitivtVerbum** , | , der **IntransitivtVerbum** , | , der **RefleksivtVerbum** sig,
Prædikat	::=	**VerbalLed Adverbial**$^{\bigcirc}$
VerbalLed	::=	**TransitivtVerbum Substantiv** | **IntransitivtVerbum** | **RefleksivtVerbum** sig
Konjunktion	::=	og | mens | hvorimod
Substantiv	::=	studenten | forelæseren | instruktoren | tutoren | Trine | programmet
TransitivtVerbum	::=	dumpede | roste | underviste | oversatte
IntransitivtVerbum	::=	forsvandt | bestod | drak | gik ned
RefleksivtVerbum	::=	kom | dummede | forberedte | morede
Adverbial	::=	ret så voldsomt | temmeligt hurtigt | ganske dårligt | helt perfekt

a) Skriv et program, der nondeterministisk genererer en sætning defineret af ovenstående grammatik. Vink: Skriv en procedure for hver kategori.

b) Argumenter for, at hvis grammatikken ændres til

Sætning ::= **Substantiv Bisætning**$^{\bigcirc}$ **Prædikat** \|
Sætning, **Konjunktion Sætning**

så vil den stadig definere de samme sætninger.

c) Vil dit program opføre sig anderledes, hvis det ændres i overensstemmelse med b)? I bekræftende fald hvordan og hvorfor?

Opgave 55

Modificer programmet i afsnit 12.4 i TRINE noten, så det beregner værdierne af de genkendte udtryk. Man skal nu benytte typen

Type Response = **Sum**(ok: Int, error: Text)

Udvid også grammatikken med en operator sqrt(**Expr**), der beregner heltalskvadratroden af sit argument.

Opgave 56

Denne opgave handler om en udvidelse af programmet i afsnit 12.4 i TRINE noten og opgave **55**. Syntaksen for *udtryk* udvides på følgende måde

Expr	::=	**Term** |
		Term + **Expr** |
		Term − **Expr**
Term	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
		(**Expr**) |
		Variable
Variable	::=	a | b | c | ...| z

En *tilstand*, der knytter heltal til variabler, kan med definitionen

Box C
    Catalog(Char, Int)
end C

repræsenteres som en værdi af typen C'Cat.

a) Skriv en procedure

Proc Expr[s: T'Seq, c: C'Cat, r: Respons]

der beregner værdien af udtrykket angivet af s relativt til c.

Syntaksen for *kommandoer* er som følger

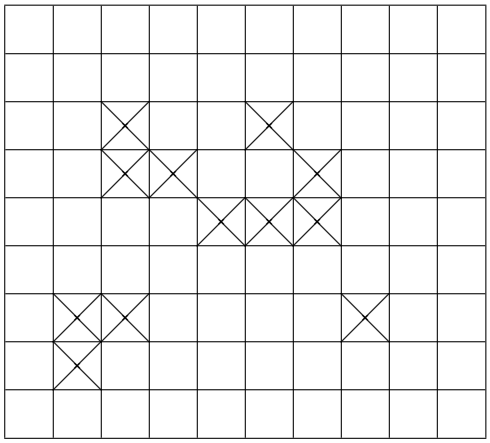
Command	::=	? **Expr** |
		! **Variable** = **Expr**

Kommandoer udføres relativt til en global tilstand. En ?-kommando beregner og udskriver værdien af udtrykket i den aktuelle tilstand. En !-kommando ændrer den aktuelle tilstand, således at variablen tilordnes værdien af udtrykket.

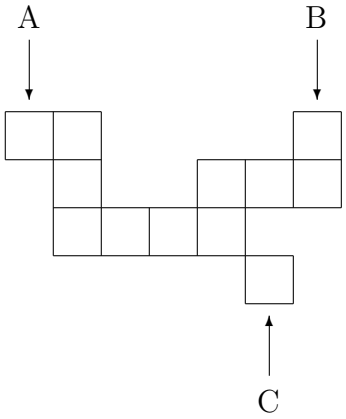
b) Skriv en proces, der fungerer som en simpel regnemaskine ved at fortolke kommandoer indtastet af brugeren.

Opgave 57

På et ternet stykke papir er ternene enten hvide eller sorte



To tern er *naboer*, hvis de har en side fælles, og to tern er *sammenhængende*, hvis de er forbundet af en følge af naboer der har samme farve. I nedenstående tegning er fx ternene A og B sammenhængende, hvorimod C ikke hænger sammen med noget.



Det ternede papir kan repræsenteres af typen

Type Farve = **Sum**(sort, hvid: Unit)
Type Papir = **List**(**List**(Farve))

Hvis p er en variabel af type Papir, så opnås et helt hvidt n×n papir med sætningen

p := List(List(Farve(hvid: #) | n) | n)

Det (i, j)'te tern gøres sort med sætningen

p.(i, j) := Farve(sort: #)

Skriv en procedure

Proc TælSorte(p: Papir, i, j: Int) → (Int)

hvis værdi er 0, hvis p.(i, j) er hvid, og som ellers angiver antallet af sorte tern, der hænger sammen med p.(i, j). Det kan antages, at papirets yderste kant er helt hvid. Vink: brug en lokal procedure.

Opgave 58

Betragt typedefinitionen

Type N_0 = **Sum**(p: N_0)

a) Find Val(N_0) og vis, hvordan den naturligt kan opfattes som mængden af ikke-negative heltal. Det vil sige, vis hvilken N_0-værdi, der skal repræsentere tallet $n \geq 0$.

b) Skriv en box

```
Box Tal
    Type N0 = Sum(p: N0)

    Proc Zero() → (N0)
        ≪returner 0≫
    end Zero

    Proc IsZero(n: N0) → (Bool)
        ≪returner n=0≫
    end IsZero

    Proc Succ(n: N0) → (N0)
        ≪returner n+1≫
    end Succ

    Proc Pred(n: N0) → (N0)
        ≪returner 0 hvis n=0, ellers n-1≫
    end Pred
end Tal
```

der bruger repræsentationen fra a) til at implementere de ikke-negative heltal. Inde i boxen må der ikke bruges variabler eller konstanter af typen Int!

c) Lav, ved hjælp af boxen Tal, en værdiprocedure

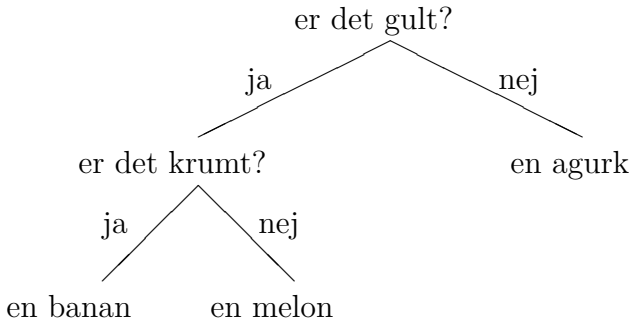
```
Proc Add(n, m: Tal'N0) → (Tal'N0)
    ≪returner n+m≫
end Add
```

d) Ville det være anderledes, hvis vi definerede

Type $N_0 =$ **Prod**(p: N_0)

Opgave 59

Et *videnstræ* er enten en *ting* eller et *spørgsmål* om ting, der kan besvares med ja eller nej sammen med to mindre videnstræer. Det følgende er et eksempel på et videnstræ



Et videnstræ kan repræsenteres med følgende rekursive type

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

a) Forklar, hvordan ovenstående træ kan være en værdi af typen Vid.

Man kan udspørge et videnstræ gennem en dialog som den følgende, hvor brugeren skriver det understregede og programmet prøver at gætte, hvilken ting man tænker på

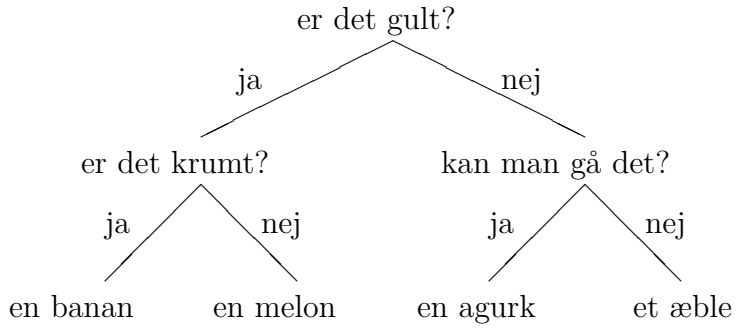
er det gult?
ja
er det krumt?
nej
er det en melon?
ja
det tænkte jeg nok!

b) Hvordan skal et videnstræ se ud, hvis programmet hurtigt skal kunne gætte, hvad man tænker på?

Hvis videnstræet ikke kender tingen, man tænkte på, så forløber dialogen på denne måde

er det gult?
nej
er det en agurk?
nej
hvad er det så?
et æble
stil et spørgsmål, der skelner en agurk fra et æble
Kan man gå det?
og hvad er svaret for en agurk?
ja

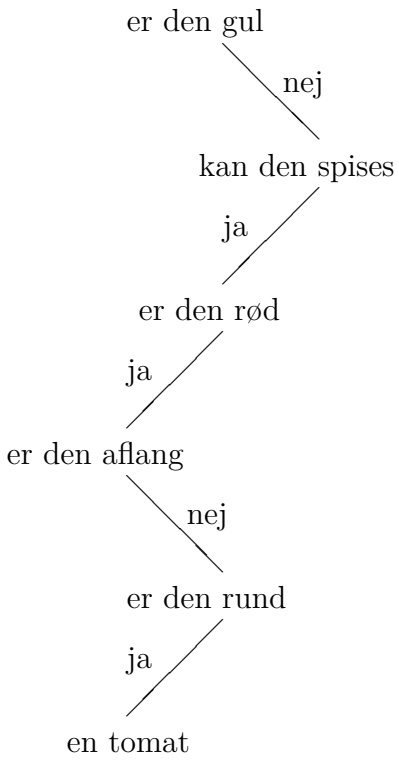
Resultatet af denne dialog er, at videnstræet nu er udvidet på følgende måde



c) Skriv et program, der realiserer dialoger som de ovenstående for videnstræer, der er repræsenterede som værdier af typen Vid. På filen `eksempler/dyr.zoo` kan man finde et større videnstræ, der handler om zoologi.

Opgave 60

Denne opgave handler om videnstræer, der blev defineret i opgave **59**. Hvis en del af et videnstræ ser ud som følger



så kan man tænke sig følgende dialog, hvor brugeren skriver det understregede

> hvad vil du gerne vide noget om? en tomat
> den er rund
> den er rød
> den kan spises

Skriv et program, der realiserer dialoger som den ovenstående for et givet videnstræ. Vink: Et spørgsmål kan (som regel) laves om til et udsagn, ved at man ombytter de to første ord.

Opgave 61

Betragt følgende rekursive type

```
Type Logisk = Sum(konst: Bool,
                  negation: Logisk,
                  konjunktion, disjunktion: LogiskListe)
Type LogiskListe = List(Logisk)
```

hvis værdier kan opfattes som logiske udtryk. Negation tager et enkelt argument, hvorimod disjunktion og konjunktion tager et vilkårligt antal argumenter (inklusive nul). Værdien af en konstant er blot dens indhold. Værdien af en negation er true hvis værdien af dens argument er false, og false hvis værdien af dens argument er true. Værdien af en disjunktion

er true hvis mindst et af dens argumenter har værdi true; ellers er værdien af disjunktionen false. Værdien af en konjunktion er false hvis mindst et af dens argumenter har værdi false; ellers er værdien af konjunktionen true. Vi har fx at værdien af

```
Logisk(konjunktion: LogiskListe(
                        Logisk(konst: true),
                        Logisk(disjunktion: LogiskListe())
                       )
       )
```

er false.

Skriv en værdiprocedure

Proc Afgør[L: Logisk] → (Bool)

der returnerer værdien af det logiske udtryk L.

Opgave 62

Vi skal se på *propositionsudtryk*, der er udtryk af følgende form

- propositioner $p_0, p_1, p_2, \ldots$
- negation $\neg P$
- konjunktion $P \wedge Q$
- disjunktion $P \vee Q$
- implikation $P \Rightarrow Q$

hvor P og Q selv er propositionsudtryk. Vi kan repræsentere propositionsudtryk som værdier af typen

Type Prop = **Sum**(pvar: Int, pnot: Prop, pand, por, pimp: Parg)
Type Parg = **Prod**(left, right: Prop)

Her kan for eksempel propositionen p_i angives som Prop(pvar: i).

a) Skriv et udtryk af type Prop, der angiver propositionsudtrykket $p_2 \vee \neg p_7$.

Et propositionsudtryk har en sandhedsværdi relativt til en *tilstand*, der angiver sandhedsværdien af de indgående propositioner. Vi kan repræsentere en tilstand som en værdi af typen

Type State = **List**(Bool)

således, at hvis S er en tilstand, så er S.(i) sandhedsværdien af p_i.

b) Skriv en procedure

Proc Eval[S: State, P: Prop] → (Bool)

der beregner sandhedsværdien af propositionsudtrykket P relativt til tilstanden S. Det kan antages, at alle variabler i P er definerede i S.

Alle propositionsudtryk kan omskrives til en ækvivalent $\neg\vee$-form, hvor kun operatorerne $\neg$ og $\vee$ bruges. For eksempel gælder det, for vilkårlige propositionsudtryk P og Q, at $P \wedge Q$ kan omskrives til $\neg(\neg P \vee \neg Q)$, og at $P \Rightarrow Q$ kan omskrives til $\neg P \vee Q$.

c) Skriv en procedure

Proc NotOr[P: Prop] → (Prop)

der beregner $\neg\vee$-formen af propositionsudtrykket P.

Opgave 63

Betragt nedenstående program. Hvad foregår der?

```
(+ Box A(T)
       Box B
           A(List(T))
       end B

       Proc Center[t: T](n: Int)
           if n = 0 →  read[t]
           | n>0 →
                 (+ Var L: List(T)
                    B'Center[L](n-1)
                    t := L.(| L |/2)
                 +)
           fi
       end Center
   end A

   Box C
       A(Int)
   end C

   Var d, i: Int

   read[d]
   C'Center[i](d)
   write(i)
+)
```

Opgave 64

Hvilke af følgende typer er ækvivalente?

```
Type A = Sum(x: B)
Type B = Sum(a: A)
Type C = B
Type D = Prod(x: A, y: C)
Type E = List(C)
Type F = List(Sum(y: A))
Type G = Sum(x: Int, y: D)
Type H = Prod(x: Int, y: D)
Type I = Prod(x, y: List(I))
Type J = List(List(J))
Type K = List(J)
```

Opgave 65

Betragt følgende program.

```
Process P
    (+ Type A = Prod(x: Int, y: B)
       Type B = List(C)
       Type C = Prod(x: D, y: List(A))
       Type D = Int
       Type E = List(D)
       Type F = List(List(Int))

       Var a: A
       Var c: C
       Var e: E
       Var f: F

       a := Prod(7, List())
       c := a
       e := List()
       f := e
    +)
end P
```

Vil det blive accepteret af oversætteren?

Opgave 66

Denne opgave omhandler labyrinterne fra afsnit 13.5 i TRINE noten.

En labyrint er *simpel*, hvis der er en vej mellem vilkårlige to steder, og der ikke er nogen løse skillevægge.

a) Vis, at programmet i afsnit 13.5 kun kan tegne simple labyrinter.

b) Vis, at programmet ikke kan tegne *alle* simple labyrinter.

c) Hvordan kunne man ændre programmet, så det *kan* tegne alle simple labyrinter?

Opgave 67

Denne opgave drejer sig om at udvide grafik fortolkeren fra kapitel 14 i TRINE noten.

Den første udvidelse er *navngivne* kommandoer. Vi vil gerne kunne skrive

```
proc Streg: drej 90; flyt 150
```

således at kommandoen Streg fremover vil svare til drej 90; flyt 150.

a) Udvid grammatikken, så den afspejler ovenstående konstruktion.

b) Udvid procedurerne Genkend og Fortolk, så den nye kommando bliver tilgængelig. Vink: Man kan med fordel benytte polyboxen Catalog til at huske de navngivne kommandoer, ved at definere

```
Box Env
    Catalog(Text,Komm)
end Env
```

Kan din fortolker klare rekursive, navngivne kommandoer?

Den anden udvidelse er med *iteration*. Vi vil have kommandoer af formen

```
gentag 4: drej 90; flyt 150
```

der vil tegne et kvadrat med sidelængde 150.

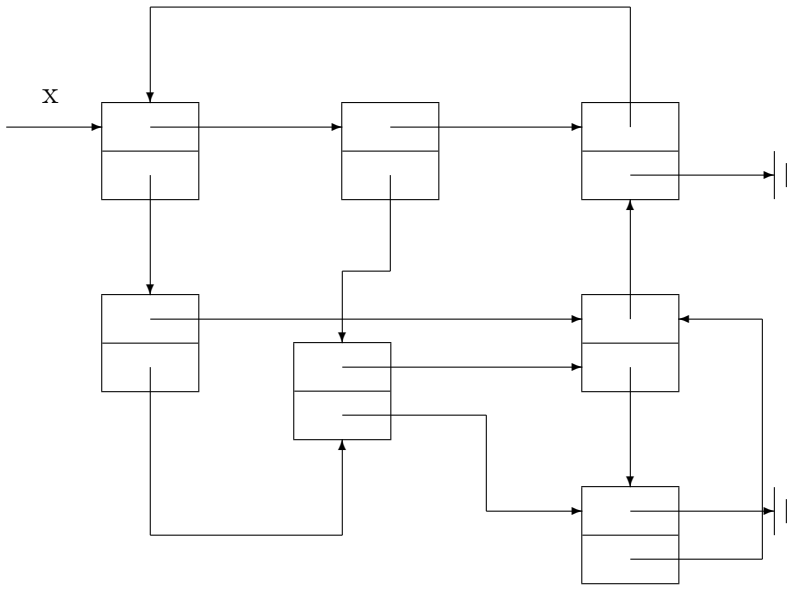
c) Gentag b) for denne nye kommando.

Opgave 68

Betragt følgende type

```
Type P = Pointer(Prod(a, b: P))
```

Det følgende billede er en "pointer-tegning" af en tilstand, der indeholder en variabel x af type P.



Angiv en tilsvarende tilstandstabel. Tegn også tilstanden efter udførslen af sætningen ref(ref(x).a).b:=x.

Opgave 69

Betragt følgende rekursive type T

Type T = **Prod**(x: S, left, right: T)
Type S = **Sum**(val: Int, extra: T)

Skriv den ved hjælp af pointere i stil med afsnit 15.3 i TRINE noten. Skriv også de tilhørende procedurer Copy, Kill, Equal og Print.

Opgave 70

Vis, ved hjælp af "pointer-tegninger",

a) at proceduren Push på side 209 i TRINE noten er korrekt

b) at procedurerne Combine og Split på side 211-212 i TRINE noten er korrekte.

Vær specielt opmærksom på specialtilfælde med korte lister af længde 0, 1 og 2.

Opgave 71

Udvid boxen Cyclic på side 208 i TRINE noten med en operation

Proc Equal[S_1, S_2: Stack] → (Bool)

der afgør, om S_1 og S_2 angiver den samme liste. Er der nogen problemer med dette?

Opgave 72

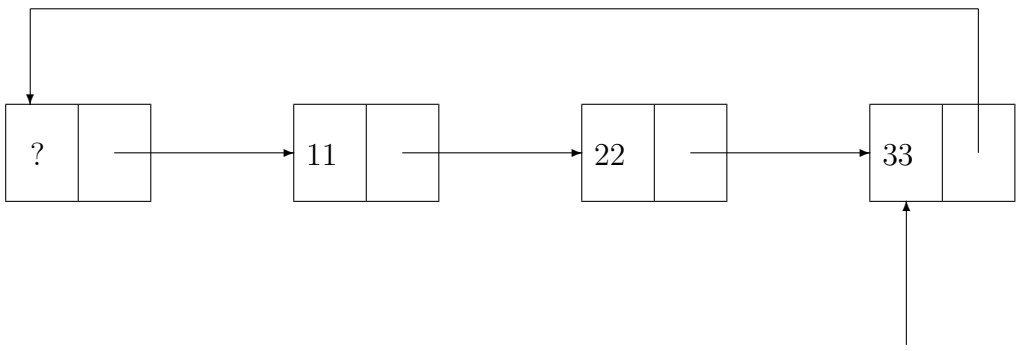
Betragt følgende datatype

```
Box Fifo
    Type Queue = ≪liste af heltal≫

    Proc Init[Q: Queue]
        ≪Q:=()≫
    end Init

    Proc Empty[Q: Queue] → (Bool)
        ≪return Q=()≫
    end Empty

    Proc Enter[Q: Queue](i: Int)
        ≪Q:=Q++(i)≫
    end Enter

    Proc Leave[Q: Queue, i: Int]
        ≪Q,i:=Q(1..|Q|),Q.(0)≫
    end Leave
end Fifo
```

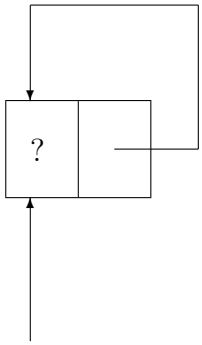
Angiv en implementation af Fifo, hvor alle operationer tager konstant tid. Typen Queue skal defineres som

```
Type Queue = Pointer(Node)
Type Node = Prod(val: Int, next: Queue)
```

hvor fx listen $(11, 22, 33)$ repræsenteres som



og den tomme liste som



Opgave 73

Udvid spalteprogrammet fra TRINE noten afsnit 16.2 til også at kunne dele ord, hvilket vil give pænere spalter.

Vink: Brug orddelingsprogrammet fra TRINE noten side 65 til at lave en ekstra process Hyp, mellem Find og Collect, der modtager ord og videresender stavelser. Ordene "forbavsende flotte spalter" vil fx blive videresendt som

for– bav– sen– de flot– te spal– ter

Modificer derefter processen Collect passende.

Opgave 74

Tegn en mulig lagerrepræsentation af tilstanden på side 48 i TRINE noten.

Opgave 75

Betragt følgende program

```
(+ Type T = List(T)

   Var x: T

   x := T()
   do true → x := T(x, x) od
+)
```

Hvad er størrelsen af værdien af x efter n gennemløb af **do**-løkken?

Opgave 76

Hvad gør følgende sætninger? Overbevis resten af øvelsesholdet om, at du har ret.

a)
```
(+ Var r, t: Int
   xio ?? r
   if ¬ (1 ≤ r ≤ 100) → abort fi
   t := 1
   do r ≤ 100 → r, t := r+11, t+1
     | t>1 → r, t := r−10, t−1
   od
   xio !! r
+)
```

b)
```
(+ Var n, i, m, r: Int
   xio ?? n
   if n<1 → abort fi
   i, m, r := 1, 0, 1
   do i ≠ n → if m>0 → m, i := m−1, i+1
              | m = 0 → r, m := r+1, r+1
              fi
   od
   xio !! r
+)
```

Opgave 77

Denne opgave går ud på at generalisere afsnit 10.6 i TRINE noten (om rekursive planter) på samme måde som afsnit 11.2 generaliserer afsnit 10.1.

a) Definer en rekursiv type Flora, der kan repræsentere en rekusiv plante.

b) Skriv to procedurer MakeFlora og DrawFlora, der skaber henholdsvis tegner en rekursiv plante.

Opgave 78

Betragt følgende dekorerede sætning, hvor x og y er variabler af typen Vector

```
{ x og y er sorterede i ikke−aftagende orden }
(+ Var i, j, k: Int
   z := Vector(0 | | x |+| y |)
   i, j, k := 0, 0, 0
   do { I }
        (i<| x |) ∧ (j<| y |) →
          if x.(i) ≤ y.(j) → z.(k), i, k := x.(i), i+1, k+1
           | x.(i) ≥ y.(j) → z.(k), j, k := y.(j), j+1, k+1
```

```
        fi
    |  (i<|x|) ∧ (j=|y|) → z.(k), i, k:=x.(i), i+1, k+1
    |  (i=|x|) ∧ (j<|y|) → z.(k), j, k:=y.(j), j+1, k+1
  od
+)
```

{ z indeholder samtlige elementer fra x og y i ikke-aftagende orden }

Definer en gyldig invariant I, så du kan argumentere for at sætningen beskriver en korrekt "fletning" af x og y.

Opgave 79

Heltalskvadratroden af et tal $n \geq 0$ er det tal $r \geq 0$, der opfylder

$$r^2 \leq n < (r+1)^2$$

Argumentér for, at følgende sætning beregner og udskriver heltalskvadratrødder

```
(+ Var n: Int
   xio ?? n
   if n<0 → abort fi
   (+ Var a, b: Int
      a, b:=0, n
      do n<b*b → b:=b-1
        | a*(a+2)<n → a:=a+1
      od
      xio !! (a+b)/2
   +)
+)
```

Opgave 80

En *intervalordbog* er en datatype, hvis værdier er *mængder* af heltal. Den har følgende operationer

$$\begin{array}{rll} \textbf{Init}[B] & \textbf{sat} & B' = \emptyset \\ \textbf{Insert}[B](x) & \textbf{sat} & B' = B \cup \{x\} \\ \textbf{Delete}[B](x) & \textbf{sat} & B' = B - \{x\} \\ \textbf{Size}[B](a,b) & \textbf{sat} & (B' = B) \ \wedge \ (\text{Size}' = |\{x \in B \mid a \leq x \leq b\}|) \end{array}$$

a) (5%) Vis, at en intervalordbog er en *udvidelse* af en almindelig ordbog, idet den sædvanlige **Member**$[B](x)$ operation kan udtrykkes ved hjælp af **Size**.

Det hævdes, at en intervalordbog kan implementeres ved hjælp af et *udvidet* søgetræ, hvor knuderne er *par* af formen

$$(x, n)$$

Her er x den sædvanlige værdi som træet er ordnet efter, og n er antallet af knuder i det undertræ som knuden er rod i. Vi har med andre ord en *stærkere* invariant end normalt. Det følgende er et eksempel på et udvidet søgetræ

$$(10, 5)$$

$$(5, 3) \qquad\qquad (14, 1)$$

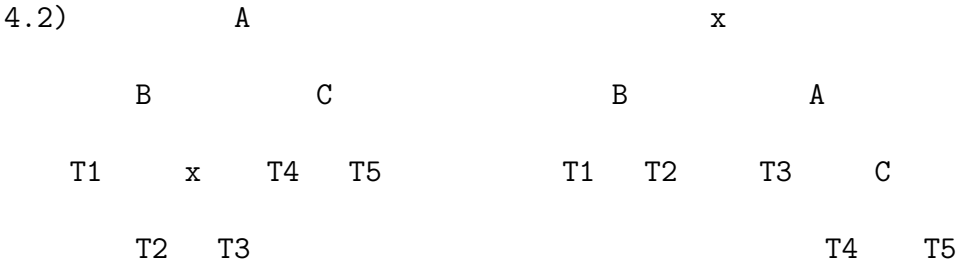
$$(3, 1) \qquad (7, 1)$$

Hvis B er denne intervalordbog, så er $\mathbf{Size}[B](6, 14)$ lig med 3.

b) (10%) Forklar, hvorledes operationerne kan implementeres, så tidskompleksiteten i alle tilfælde bliver proportional med højden af det udvidede søgetræ. Vink: Bemærk, at

$$|\{x \in B \,|\, a \leq x \leq b\}| = |B| - |\{x \in B \,|\, x < a\}| - |\{x \in B \,|\, x > b\}|$$

Som sædvanligt er det ønskeligt, at træet bliver balanceret; det vil sige, at dets højde skal være proportional med *logaritmen* til antallet af knuder. En måde at opnå dette på er som bekendt at benytte den *rød/sorte* strategi, der blandt andet omfatter følgende transition



for indsættelser.

c) (5%) Angiv, hvordan ovenstående transition skal modificeres til også at virke for udvidede søgetræer.

Vi indfører nu en ekstra operation, **Find**, der giver det k'te element i ordbogen. Specifikationen er

$$\mathbf{Find}[B](k) \ \mathbf{sat} \ (1 \leq k \leq |B|) \ \rightarrow (B' = B) \ \wedge \ (Find' \in B) \wedge (|\{x \in B \mid x \leq \mathrm{Find}'\}| = k)$$

d) (5%) Hvordan kan man implementere **Find** operationen? Hvad bliver tidskompleksiteten?

Opgave 81

Denne opgave handler om at finde *inversioner* i en vektor. En inversion er et par af elementer, der står forkert i forhold til hinanden, hvis vektoren skulle være sorteret i sædvanlig ikke-aftagende orden. I følgende eksempel

$$W = \boxed{1}\boxed{2}\boxed{3}\boxed{0}\boxed{4}\boxed{2}\boxed{5}\boxed{5}\boxed{6}\boxed{3}\boxed{7}$$

udgør de to med pile angivne elementer således en inversion blandt flere. Formelt definerer vi en inversion i en vektor V som et par af *indices* (i, j) hvorom der gælder, at

$$(i < j) \wedge (V.(i) > V.(j))$$

Vi er interesserede i at finde "det største antal inversioner" i en sådan vektor V – mere præcist ønsker vi at finde

$$\max_{0 \le i < |V|} Inv(V, i)$$

hvor

$$Inv(V, i) = |\{j \mid (i < j < |V|) \wedge (V.(i) > V.(j))\}|$$

Det vil sige, at vi skal finde det største antal inversioner som noget enkelt element er skyld i. Det er nemt at konstruere en $O(|V|^2)$-algoritme, der løser problemet, idet $Inv(V, i)$ kan beregnes i tid $O(|V|-i)$. Problemet kan imidlertid også løses mere effektivt, og til den ende introduceres følgende definition: En *spærring* i en vektor V er et indeks s, for hvilket $V.(s)$ er større end alle elementer i $V(0\,..\,s-1)$. Mere præcist er s en spærring, hvis der gælder

$$(0 \le s < |V|) \wedge (\forall j : (0 \le j < s) : V.(j) < V.(s))$$

a) (5%) Angiv funktionen $Inv(W, i)$ samt spærringerne i vektoren W ovenfor ved at udfylde resten af følgende tabel

spærring	i	$Inv(W, i)$
$*$	0	1
	1	
	2	
	$\vdots$	
	9	
	10	

hvor $*$ markerer, at det pågældende indeks er en spærring i W. Vis også, at hvis $s_0 < s_1 < \cdots < s_{m-1}$ udgør spærringerne i en vektor V $(m \le |V|)$ så gælder der, at

- $V.(s_0) < V.(s_1) < \cdots < V.(s_{m-1})$
- $\max_{0 \le i < |V|} Inv(V, i) = \max_{0 \le j < m} Inv(V, s_j)$

I det følgende betegnes den voksende følge af spærringer i en vektor V med *spær*(V). Betragt følgende algoritme:

Algoritme: Spærringer
Stimulans : V: Vector, $|\mathrm{V}| > 0$
Respons : Sp: Vector, Sp = *spær*(V)
Metode : ≪Initialiser Sp, m, max og i≫
do $\{(\mathrm{Sp}(0\,..\,\mathrm{m}-1) = \mathit{spær}(\mathrm{V}(0\,..\,\mathrm{i}-1))) \wedge (\mathrm{max} \geq \mathrm{V}(0\,..\,\mathrm{i}-1))\}$
$\mathrm{i} \neq |\,\mathrm{V}\,| \rightarrow$ ≪Opdater Sp, m og max≫
i:=i+1
od
Sp:=Sp(0..m-1)

b) (10%) Konkretiser algoritmen Spærringer, så den bliver gyldig og korrekt. Bevis korrektheden.

Vi kalder nu et indeks s for en *halv*spærring, hvis $V.(s)$ er større end *eller lig med* $\mathrm{V}.(0\,..\,s-1)$. Vi er interesserede i følgende vektor $\mathrm{M}(0\,..\,|\mathrm{V}|-1)$ hvor

$$\mathrm{M}.(i) = \begin{cases} -1, \text{ hvis } i \text{ er en halvspærring} \\ \text{mindste } k \text{ så } \mathrm{V}.(\mathrm{Sp}(k)) > \mathrm{V}.(i), \text{ ellers} \end{cases}$$

c) (5%) Forklar i (få) ord hvordan man kan beregne M. Hvad bliver tidskompleksiteten?

Betragt følgende algoritme:

Algoritme: Inversioner
Stimulans : V:Vector, $|\mathrm{V}| > 0$
Respons : maxinv: maxinv = $\max_{0 \leq i < |V|} \mathit{Inv}(\mathrm{V}, i)$
Metode : ≪Udfør algoritme Spærringer≫
≪Konstruer vektoren M≫
i, Tæl, t, k, maxinv:=|V|, Vector(0||Sp|-1), 0, |Sp|-1, 0
do {I}
$\mathrm{i} \neq 0 \rightarrow$ i:=i-1
if M.(i) < 0 →
if i=Sp.(k) →
maxinv:=max{maxinv, t}
t, k:=t-Tæl.(k), k-1
fi
| M.(i) ≥ 0 →
Tæl.(M.(i)):=Tæl.(M.(i))+1
t:=t+1
fi
od

hvor invarianten er givet ved

$$
\begin{aligned}
\mathrm{I} \;:\; & (\mathrm{maxinv} = \max_{\mathrm{i} \le j < |V|} \mathit{Inv}(\mathrm{V}, j)) \wedge \\
& (\forall l : (0 \le l < \mathrm{m}) : \mathrm{Tæl.}(l) = |\{j : (\mathrm{i} \le j < |\mathrm{V}|) \mid \mathrm{M.}(j) = l\}|) \wedge \\
& (t = |\{j : (\mathrm{i} \le j < |\mathrm{V}|) \mid \mathrm{M.}(j) \le \mathrm{k}\}|)
\end{aligned}
$$

d) (5%) Forklar hvad denne algoritme foretager sig og gør herunder rede for dens korrekthed. Hvad bliver tidskompleksiteten?

Opgave 82

I denne opgave skal der skrives en TRINE box, der implementerer en lille del af RASMUS.

Et *skema* repræsenteres med følgende type

```
Type AttName = Text
Type Schema = List(AttName)
```

Dette skema rummer kun navnene på attributterne; i modsætning til RASMUS er rækkefølgen af attributterne signifikant. Vi vedtager, at alle attributter skal have type Text.

Et *tupel* kan repræsenteres med følgende type

```
Type AttValue = Text
Type Tuple = List(AttValue)
```

En samling tupler kan repræsenteres som en sekvens

```
Box T
    Sequence(Tuple)
end T
```

Der skal nu skrives en box

```
Box Ras
    Type Relation = Prod(s: Schema, t: T'Seq)

    Proc Print[R: Relation]
        ≪udskriv R på skærmen≫
    end Print

    Proc Union[R1, R2: Relation] → (Relation)
        ≪beregn foreningen af R1 og R2≫
    end Union

    Proc Project[R: Relation](p: Schema) → (Relation)
```

```
        ≪projicer R på p≫
    end Project

    Proc Join[R1, R2: Relation] → (Relation)
        ≪beregn join af R1 og R2≫
    end Join

    @"c:\dat1\read.tri"

end Ras
```

der implementerer en datatype af relationer.

- Start med at beskrive præcist, hvad resultaterne af Union, Project og Join skal være.
- Procedurerne Union og Project skal abortere, hvis parametrene ikke er lovlige.
- Parameteren p af type Schema i proceduren Project angiver de attributnavne, der skal beholdes.
- Filen `c:\dat1\read.tri` indeholder en procedure

  ```
  Proc Read[R: Relation](f: Text)
  ```

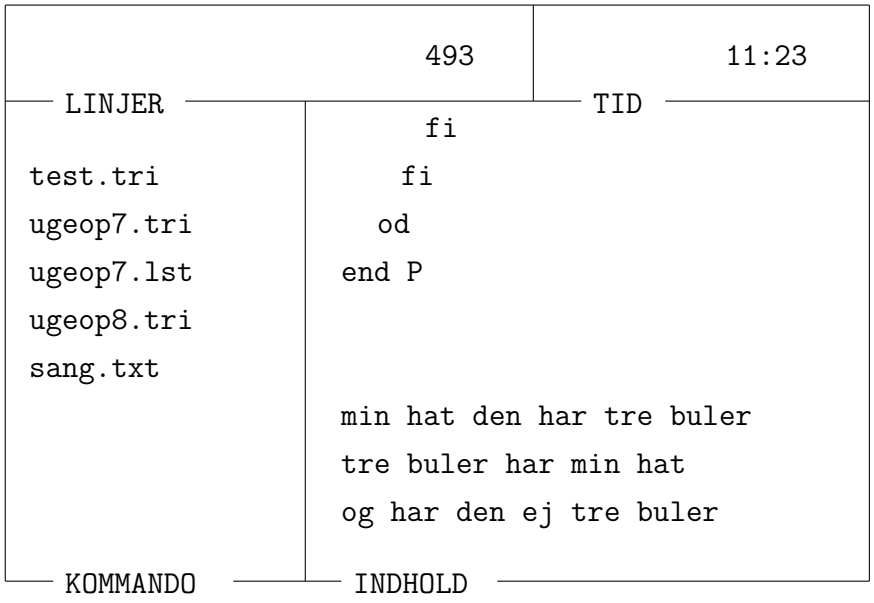
  der fra filen med navn f indlæser en RASMUS relation, der skal være udskrevet i RASMUS's `write-to` format (beskrevet i RASMUS noten side 63).
- Filen `c:\dat1\filter.tri` indeholder en procedure

  ```
  Proc Filter[t: T'Seq]
  ```

  der fjerner alle dubletter i en sekvens af tupler; man kan frit gøre brug af denne.
- Ovenstående skitse ligger på filen `c:\dat1\ras.tri`.
- Vink til Project: Beregn først indices på de attributter, der skal bevares.
- Vink til Join: Beregn først de par af indices på attributter, der skal være ens, for at to tupler kan sættes sammen. Beregn også indices på de overskydende attributter i R_2.

Opgave 83

Skriv et system, der implementerer fire vinduer på skærmen med følgende udseende



I vinduet `TID` vises løbende hvor lang tid systemet har kørt. I vinduet `KOMMANDO` kan man skrive navne på DOS-filer, hvis indhold så vil blive vist i vinduet `INDHOLD`; disse to vinduer fungerer asynkront. Hvis man skriver en tom linje i `KOMMANDO`, så vil uret i `TID` blive nulstillet. I vinduet `LINJER` kan man løbende se, hvor mange linjer der alt i alt er blevet vist i `INDHOLD`.

Vink: Man skal have en proces for hvert vindue, en timer process og en buffer kanal.

Opgave 84

En *multiordbog* er en datatype, hvis værdier er *multimængder* af heltal, det vil sige samlinger i hvilke et tal gerne må optræde flere gange. En multiordbog M kan opfattes som en funktion fra heltal til heltal, således at $M(x)$ angiver antallet af forekomster af x. Der er følgende operationer

$$
\begin{array}{rll}
\textbf{Init}[M] & \textbf{sat} & \forall x:\ M'(x) = 0 \\
\textbf{Insert}[M](x) & \textbf{sat} & (M'(x) = M(x){+}1) \wedge \\
 & & (\forall y : (y \neq x) : M'(y) = M(y)) \\
\textbf{Member}[M](x) & \textbf{sat} & (M' = M) \wedge (\mathrm{Member}' = (M(x) > 0)) \\
\textbf{Delete}[M](x) & \textbf{sat} & (M'(x) = max\{0, M(x){-}1\}) \wedge \\
 & & (\forall y : (y \neq x) : M'(y) = M(y))
\end{array}
$$

a) (5%) Beskriv en datastruktur, der implementerer en multiordbog med følgende worst-case tidskompleksiteter

$$\begin{aligned} T[\![\mathbf{Init}[M]]\!] &\in O(1) \\ T[\![\mathbf{Insert}[M](x)]\!] &\in O(\log \|M\|) \\ T[\![\mathbf{Member}[M](x)]\!] &\in O(\log \|M\|) \\ T[\![\mathbf{Delete}[M](x)]\!] &\in O(\log \|M\|) \end{aligned}$$

hvor $\|M\|$ er antallet af *forskellige* elementer i M.

Multiordbogen kan udvides med følgende operation

$$\begin{aligned} \mathbf{MaxFreq}[M] \quad \mathbf{sat} \quad & \|M\| > 0 \rightarrow \\ & (M' = M) \wedge (\forall x : \ (M(x) \leq M(\mathrm{MaxFreq}'(\mathrm{MaxFreq}'))) \end{aligned}$$

det vil sige **MaxFreq** returnerer det hyppigst forekommende element; hvis to forskellige elementer begge forekommer med maximal hyppighed, så er det ligegyldigt, hvilket der vælges.

b) (10%) Beskriv en datastruktur for den udvidede multiordbog med worst-case tidskompleksiteter

$$\begin{aligned} T[\![\mathbf{Init}[M]]\!] &\in O(1) \\ T[\![\mathbf{Insert}[M](x)]\!] &\in O(\log \|M\|) \\ T[\![\mathbf{Member}[M](x)]\!] &\in O(\log \|M\|) \\ T[\![\mathbf{Delete}[M](x)]\!] &\in O(\log \|M\|) \\ T[\![\mathbf{MaxFreq}[M])]\!] &\in O(\log \|M\|) \end{aligned}$$

I stedet for **MaxFreq** kunne man vælge at have en operation **Freq**$[M](k)$, der returnerede det k'te hyppigste element i M.

c) (10%) Angiv en formel specifikation for operationen **Freq** og en mulig implementation. Hvilken tidskompleksitet får denne?

Opgave 85

Betragt profilerne for n-terninger fra side 149 i TRINE noten.

$$\begin{array}{lccccc} 0: & 1 & & & & \\ 1: & 2 & 1 & & & \\ 2: & 4 & 4 & 1 & & \\ 3: & 8 & 12 & 6 & 1 & \\ 4: & 16 & 32 & 24 & 8 & 1 \\ \vdots & & & & & \end{array}$$

Hvis en del af ovenstående trekant ser ud som

$$\begin{array}{cc} A & B \\ & C \end{array}$$

så gælder det, at $C = 2B + A$.

a) Giv på denne baggrund en rekursiv definition af

$$\left[\begin{array}{c} n \\ k \end{array} \right]$$

der er det k'te element i en n-ternings profil, for $0 \leq k \leq n$. Vink: Dette minder *meget* om binomialkoefficienter.

b) Skriv en rekursiv værdiprocedure

Proc Prof(n, k: Int) → (Int)

der beregner disse værdier.

c) Tegn udførelsestræet (med rekursive instanser i knuderne) for kaldet

Prof(4, 2)

Angiv tilstandstabellerne, der svarer til forskellige veje i dette træ.

Opgave 86

En tekst er som bekendt et *palindrom*, hvis den er lig med sin egen spejling, det vil sige, hvis den opfylder prædikatet

$$pal(t) : \ (\forall i \in 0..|t| : \ t.(i) = t.(|t|-i-1))$$

Man kan nemt afgøre om t er et palindrom i tid $O(|t|)$.

Angiv en algoritme k-pal, der afgør om en tekst t har en deltekst af længde k, der er et palindrom. Angiv tidskompleksiteten $T[\![k\text{-pal}]\!](|t|, k)$.

Angiv også en algoritme, der afgør om t har en deltekst af en eller anden længde ≥ 2, der er et palindrom. Hvad er den bedste tidskompleksitet, du kan opnå?

Opgave 87

En operation f på *to* argumenter er

- *associativ*, hvis $f(x, f(y, z)) = f(f(x, y), z)$ for alle x, y, z
- *kommutativ*, hvis $f(x, y) = f(y, x)$ for alle x, y
- *absorptiv*, hvis $f(f(x, y), y) = f(x, y)$ for alle x, y

En operation g på *et* argument er

- *idempotent*, hvis $g(g(x)) = g(x)$ for alle x
- *monoton*, hvis $x \leq y$ medfører $g(x) \leq g(y)$

Hvilke RASMUS operationer er associative, kommutative, absorptive, idempotente eller monotone?

Opgave 88

Denne opgave undersøger egenskaber ved **join** operationen i RASMUS.

a) Hvad kan man sige om et **join** af to relationer uden fælles attributnavne?

b) Hvad kan man sige om et **join** af to relationer med samme skema?

c) Hvad er sammenhængen mellem R og

join
    project R **over** S_1
with
    project R **over** S_2

hvor S_1 og S_2 er en opdeling af R's skema?

Opgave 89

Denne opgave undersøger ligheder mellem RASMUS-relationer og de hele tal. Vi skriver **join** som $*$ og **union** som $+$.

a) Eftervis, at følgende (velkendte) regler gælder.

$$
\begin{aligned}
&R_1 + R_2 = R_2 + R_1 \\
&R_1 + (R_2 + R_3) = (R_1 + R_2) + R_3 \\
&R_1 * R_2 = R_2 * R_1 \\
&R_1 * (R_2 * R_3) = (R_1 * R_2) * R_3 \\
&R_1 * (R_2 + R_3) = R_1 * R_2 + R_1 * R_3
\end{aligned}
$$

b) Hvilke relationer svarer til 0 og 1, sådan at følgende regler også gælder?

$$0 + R = R$$
$$1 * R = R$$
$$0 * R = 0$$

c) Kan man også finde en operation, der svarer til (heltals)division?

Opgave 90

Betragt følgende rekursive type T

Type T = **Prod**(x: S, left, right: T)
Type S = **Sum**(val: Int, extra: T)

Skriv den ved hjælp af pointere i stil med afsnit 15.3 i TRINE noten. Skriv også de tilhørende procedurer Copy, Kill, Equal og Print.

Opgave 91

Følgende algoritme repræsenterer skolealgoritmen til multiplikation af heltal, jfr. også [Algoritmisk Problemløsning] afsnit 1.6.

Algoritme: Multiplikation
Stimulans: $x, y \geq 0$
Respons: $z = x * y$
Metode: $z, i := 0, 0$
do $\{I\}$
$i < ||y|| \rightarrow$
$z := z + ((x \otimes y_i) \uparrow i)$
$i := i + 1$
od

Her angiver $||y||$ antallet af cifre i y, y_i angiver det i'te ciffer i y, $\otimes$ betyder multiplikation af et tal med et enkelt ciffer, og $h \uparrow i$ betyder $h * 10^i$.

Angiv en gyldig invariant I for algoritmen, som er nyttig i forbindelse med et korrekthedsbevis (det kræves ikke, at beviset gennemføres).

Algoritmen Multiplikation kan omskrives til en procedure på følgende måde

Proc Mult(x, y: Int) → (Int)
(+ **Var** z, i: Int
$z, i := 0, 0$
do $\{I\}$

```
            i < ||y|| →
                z :=z + ((x ⊗ y_i) ↑ i)
                i :=i + 1
        od
        return z
    +)
end Mult
```

Angiv en specifikation af Mult. Bevis dernæst, at nedenstående algoritme er gyldig og korrekt.

Algoritme: Itereret kvadrering
Stimulans: $x, n \geq 0$
Respons: $r = x^{2^n}$
Metode:
```
r, i :=x, 0
do {(r = x^(2^i)) ∧ (0 ≤ i ≤ n)}
   i < n →
      r :=Mult(r,r)
      i :=i + 1
od
```

Opgave 92

Udvid spalteprogrammet fra TRINE noten afsnit 16.2 til også at kunne dele ord, hvilket vil give pænere spalter.

Vink: Brug orddelingsprogrammet fra TRINE noten side 65 til at lave en ekstra process Hyp, mellem Find og Collect, der modtager ord og videresender stavelser. Ordene "forbavsende flotte spalter" vil fx blive videresendt som

for– bav– sen– de flot– te spal– ter

Modificer derefter processen Collect passende.

Opgave 93

Der ønskes en TRINE-sætning, der kan indlæse tal mellem 0 og 99 og udskrive deres tekstuelle repræsentation. Fx skal 3 blive til "tre" og 47 til "syvogfyrre". Sætningen kunne se ud som

```
(+ Var t: Tabel
   Var n: Int
   Var r: text
   ≪ initialiser t ≫
   xio ?? n
   do 0 ≤ n < 100 →
```

```
            ≪ omskriv n vha. t til r ≫
            xio !! r, eol
            xio ?? n
        od
    +)
```

Tabel er her en type beskrevet ved

```
Type Page = List(Text)
Type Tabel = Prod(små, mellem, store: Page)
```

og ≪ initialiser t ≫ kunne med fordel skrives i stil med

```
t := Tabel(Page("" | 10), Page("" | 10), Page("" | 10))
t.små.(0) := "nul"
⋮
t.små.(9) := "ni"
t.mellem.(0) := "ti"
⋮
t.mellem.(9) := "nitten"
t.store.(2) := "tyve"
⋮
t.store.(9) := "halvfems"
```

Opgave 94

En ny slags *udtrykstræer* med heltalsværdier defineres rekursivt som følger

- Et udtrykstræ kan være et blad indeholdende et heltal. Værdien af dette udtrykstræ er indholdet af bladet.
- Et udtrykstræ kan være af formen

  $$+$$

  $$T_1 \quad \cdots \quad T_k$$

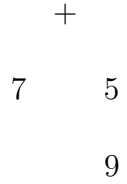
  hvor $k \geq 0$ og T_i'erne er udtrykstræer. Hvis $k = 0$ så er værdien 0; ellers er værdien summen af værdierne af T_i'erne.
- Et udtrykstræ kan være af formen

  $$m$$

  $$T$$

  hvor m er et heltal og T er et udtrykstræ. Værdien af dette udtrykstræ er m gange værdien af T.

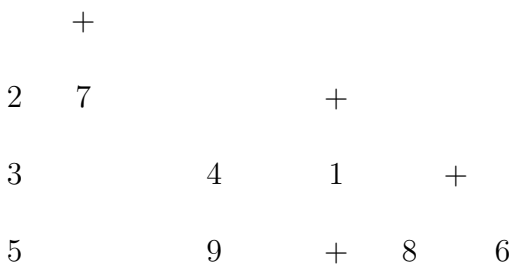
For eksempel har udtrykstræet



værdien 52. Vi kan repræsentere udtrykstræer som værdier af typen Udtryk, der defineres som følger

Type Udtryk = **Sum**(blad: Int, plus: pUdtryk, gange: gUdtryk)
Type pUdtryk = **List**(Udtryk)
Type gUdtryk = **Prod**(m: Int, faktor: Udtryk)

a) (10%) Beregn værdien af udtrykstræet

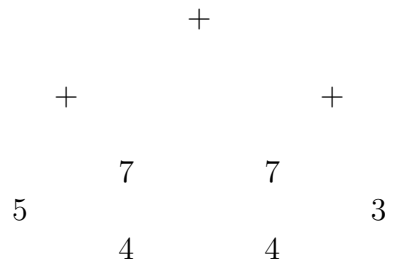


og skriv en procedure

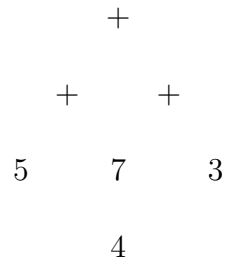
Proc Beregn[u: Udtryk] → (Int)

der beregner værdien af et vilkårligt udtrykstræ u.

Hvis et udtrykstræ har flere ens deltræer, så kan man spare plads ved at repræsentere det som en acyklisk orienteret graf, som i følgende eksempel, hvor udtrykstræet



kan ses at have to ens undertræer. Dette træ kan repræsenteres som følgende udtryksgraf



b) (5%) Skriv en type, Graf, hvis værdier kan repræsentere udtryksgrafer. Vis, hvordan eksempelgrafen repræsenteres.

Denne repræsentation kan også udnyttes til at spare tid.

c) (10%) Skriv en TRINE procedure

Proc Hurtig[g: Graf] → (Int)

der beregner værdien af udtryksgrafen g *uden* at beregne værdien affælles deludtryk mere end én gang.

Opgave 95

I denne opgave betragter vi to vektorer F og A med samme længde. Vektorerne er *konsistente*, skrevet $kons(F, A)$, hvis

$$\forall i, j\ (0 \leq i, j < |F|) :\ (F.(i) = F.(j)) \Rightarrow (A.(i) = A.(j))$$

Vi kræver altså, at indekser med samme værdi i F også skal have samme værdi i A. For eksempel har vi, at

$$F = (2, 8, 5, 2, 5, 7, 8) \quad \text{og} \quad A = (1, 7, 6, 1, 6, 1, 7)$$

er konsistente, hvorimod

$$F = (1, 1, 3, 1, 3, 1) \quad \text{og} \quad A = (4, 4, 5, 2, 5, 2)$$

ikke er konsistente. Vi skal afgøre konsistens under forskellige antagelser om F.

Vi antager først, at F er *sorteret*, det vil sige

$$\forall i, j\ (0 \leq i \leq j < |F|) :\ F.(i) \leq F.(j)$$

I så fald kan vi bruge følgende algoritme

Algoritme: K1

Stimulans	:	F, A: Vector, $\|F\|=\|A\|$, F sorteret
Respons	:	R: Bool, R=*kons*(F, A)
Metode	:	R, k := true, 1
		do k $<\|$ F $\|\rightarrow$
		if $F.$(k-1)$=F.$(k) $\rightarrow$ R := R $\wedge$ $(A.$(k-1)$=A.$(k)$)$ **fi**
		k := k+1
		od

der har tidskompleksitet $O(|F|)$.

a) (10%) Bevis, at algoritme K1 er korrekt.

Nu må vi ikke længere antage, at F er sorteret. I stedet for antager vi, at F er *begrænset*, det vil sige

$$\forall i\ (0 \leq i < |F|) :\ 0 \leq F.(i) < |F|$$

Vi skal skrive en ny algoritme

Algoritme: K2

Stimulans	:	F, A: Vector, $\|F\|=\|A\|$, F begrænset
Respons	:	R: Bool, R=*kons*(F, A)
Metode	:	$\cdots$

der løser problemet under de ændrede forudsætninger.

b) (10%) Skriv i detaljer en metode, så algoritme K2 bliver korrekt og har tidskompleksitet $O(|F|)$. Der skal argumenteres for løsningens korrekthed, men det er ikke nødvendigt med et formelt bevis.

Til sidst skal vi løse det helt generelle problem, hvor må vi ikke gøre nogen antagelser om F.

c) (5%) Skitsér en algoritme, K3, der løser problemet med tidskompleksitet $O(|F| \log |F|)$.