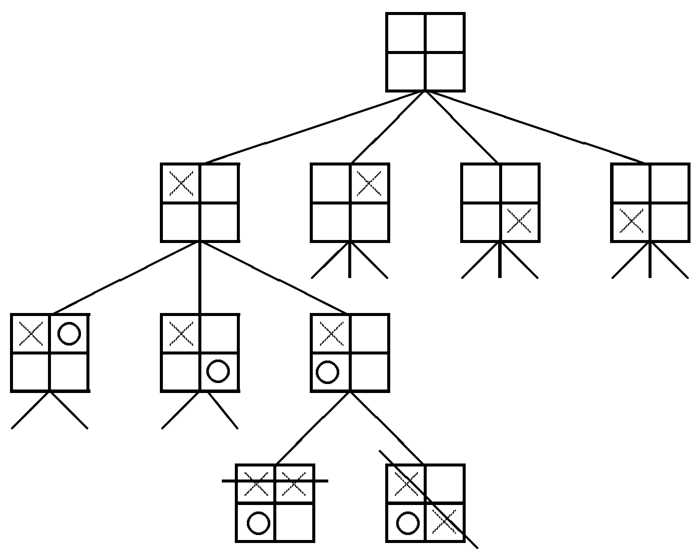
Opgave 1 (20%)

To personer A og B spiller et spil, i hvilket de skiftes til at foretage et træk. Et spil ender altid enten med uafgjort eller med at A eller B vinder. Samtlige mulige spil kan beskrives med det såkaldte *spilletræ*, der er en bestemt værdi af typen Spil:

> **Type** Stilling = ≪en stilling i spillet≫
> **Type** Spil = **Prod** (s: Stilling, t: Træk)
> **Type** Træk = **List** (Spil)

Et træ indeholder den aktuelle stilling og for samtlige mulige træk de undertræer, disse fører til. Dette fortsætter indtil man når et blad, hvor spillet altid er slut. Fx vil en del af spilletræet for det lidt kedelige spil 2×2 kryds-og-bolle se således ud:



Vi er givet tre værdiprocedurer:

> **Proc** A (s: Stilling) → (Bool)
> **Proc** B (s: Stilling) → (Bool)
> **Proc** U (s: Stilling) → (Bool)

der afgør om en given stilling er henholdsvis vundet af A, vundet af B eller endt uafgjort. Bemærk, at disse tre muligheder er gensidigt udelukkende, og at et blad altid vil opfylde en af dem.

Vi er interesserede i at afgøre, om A altid kan vinde spillet, hvis han får lov til at starte. Det gælder fx for 2×2 kryds-og-bolle, men ikke for 3×3 kryds-og-bolle, der jo kan ende uafgjort. Denne egenskab kan formuleres på følgende gensidigt rekursive facon:

- A er sikker vinder, hvis han enten har vundet roden af spilletræet eller kan foretage et træk, der fører til et spilletræ, hvor B er sikker taber.
- B er sikker taber, hvis han enten har tabt roden af spilletræet, eller kun kan foretage træk, der fører til spilletræer, hvor A er sikker vinder.

> Skriv to gensidigt rekursive TRINE værdiprocedurer:
>
> **Proc** Avinder[s: Spil] $\rightarrow$ (Bool)
> **Proc** Btaber[s: Spil] $\rightarrow$ (Bool)
>
> der implementerer denne analyse. Der lægges vægt på, at besvarelsen er letlæselig, detaljeret og korrekt.

Opgave 2 (20%)

Heltalskvadratroden af et ikke-negativt heltal n er som bekendt det tal q, for hvilket:

$$q^2 \le n < (q+1)^2$$

Følgende algoritme vides at være korrekt:

Algoritme: Heltalskvadratrod
Stimulans: n: $n \ge 0$
Respons: q: $q^2 \le n < (q+1)^2$
Metode: q,r := 0, n
do $\{ (n = q^2 + r) \wedge (r \ge 0) \}$
  r $\ge$ 2*q+1 $\rightarrow$ q, r := q+1, r-(2*q+1)
od

> **a)** Angiv algoritmens udførelsestid.

Betragt følgende alternative algoritme:

Algoritme: Heltalskvadratrod
Stimulans: n: $n \ge 0$
Respons: q: $q^2 \le n < (q+1)^2$
Metode: q,r := 0, n
do $\{ I \}$
  r $\ge$ 2*q+1 $\rightarrow$
    $\ll$find x så $(x \ge 1) \wedge (2qx + x^2 \le r)\gg$
    q, r := q+x, r-(2*q*x+x*x)
od

> **b)** Find en passende invariant og bevis at algoritmen er korrekt.

> **c)** Find en konkretisering af $\ll$find x så ... $\gg$, så udførelsestiden bliver væsentlig bedre end i spørgsmål a). Den forbedrede udførelsestid skal angives, men der kræves ikke noget bevis.

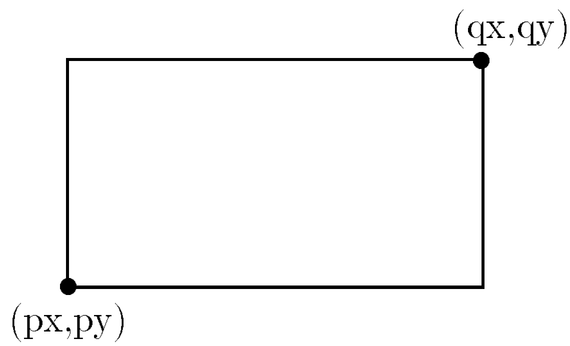
Opgave 3 (20%)

En datastruktur G skal indeholde punkter i planen, givet ved deres (heltallige) (x,y)-koordinater. I første omgang skal datastrukturen understøtte operationerne:

> **Init** [G]
> **Insert** [G] (x,y)
> **Delete** [G] (x,y)
> **Member** [G] (x,y)

a) Angiv en implementation, så alle udførelsestiderne tilhører $O(\log n)$, hvor n er antallet af punkter i strukturen G.

Et *rektangel* i planen angives ved to punkter, (px,py) og (qx,qy), der angiver henholdsvis det sydvestlige og nordøstlige hjørne:



Vi ønsker at tilføje operationen:

> **Contained** [G] (px,py,qx,qy)

som udskriver samtlige de punkter i datastrukturen, der er indeholdt i rektanglet.

b) Angiv en hensigtsmæssig implementation af den udvidede datastruktur og angiv udførelsestiderne for operationerne.

Opgave 4 (20%)

I denne opgave indfører vi en ny operator i RASMUS. Syntaksen er:

```
R $ a,b
```

hvor `a` og `b` er navne på attributter i relationen `R`. Resultatet er en delrelation af `R |+ a,b`, hvis tupler intuitivt er de, hvis `a`-attribut på flest måder kan forlænges til tupler i `R`. I det følgende bruger vi denne eksempelrelation:

a:Int	b:Int	c:Int
10	20	1
10	20	2
10	20	3
11	20	4
10	30	4
10	87	1
10	87	2
10	87	3
11	87	4
11	87	5
11	87	6

Formelt bestemmes tuplerne i `R $ a,b` som følger. Lad $\boxed{\alpha \mid \beta}$ være et tupel i `R |+ a,b`. Vi definerer nu $N(\alpha,\beta)$ som antallet af forskellige tupler $\boxed{\tau}$ i `R |- a,b` således at $\boxed{\alpha \mid \beta \mid \tau}$ er et tupel i `R`.

a) Angiv for eksempelrelationen værdierne af $N(10,20)$, $N(11,20)$, $N(10,30)$, $N(10,87)$ og $N(11,87)$.

For alle tupler $\boxed{\beta}$ i `R |+ b` indeholder `R $ a,b` det eller de tupler $\boxed{\alpha}\boxed{\beta}$, der maksimerer $N(\alpha,\beta)$.

b) Angiv resultatet af at beregne `R $ a,b` på denne relation.

c) Angiv, hvilke af følgende regneregler, der er gyldige (under antagelse af at begge sider er lovlige relationsudtryk). Begrund dine svar.

1) `R $ a,b = R $ b,a`

2) `(R + S) $ a,b = (R $ a,b) + (S $ a,b)`

3) `(R $ a,b) $ a,b = R $ a,b`

d) Vis, hvorledes `R $ a,b` kan udtrykkes ved hjælp af de sædvanlige operatorer i RASMUS.

Opgave 5 (20%)

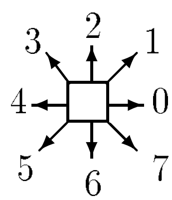
Et kvadratisk stykke ternet papir, hvor hvert felt enten er blankt eller indeholder et kryds eller en bolle, kan repræsenteres som en værdi af følgende type:

> **Type** Felt = **Sum** (kryds, bolle, blank: Unit)
> **Type** Papir = **List** (**List** (Felt))

Vi er interesserede i for hvert felt at tælle, hvormange krydser på stribe der maksimalt skærer gennem feltet (lodret, vandret eller diagonalt). Svaret kan repræsenteres som en værdi af typen:

> **Type** Xtabel = **List** (**List** (Int))

Det er bekvemt at nummerere de forskellige retninger som følger:



Det følgende er et eksempel på et stykke papir og det ønskede svar:

×	×	○			×		○	×	×
○		×	○	○	×	○		×	○
		○	×				×	○	○
○	○	×				×	○	×	○
×	×	×	○		×		×	×	○
○	○	×		×	○	×			×
○		×		○	×				
	×	×		×	○		○	○	×
	○	×			○	×		○	○
○		×	×	○	×	○	○	×	○

2	3	0	0	0	2	0	0	2	6
0	0	3	0	0	2	0	0	6	0
0	0	0	3	0	0	0	6	0	0
0	0	7	0	0	0	6	0	5	0
3	3	7	0	0	6	0	5	2	0
0	0	7	0	6	0	5	0	0	2
0	0	7	0	0	5	0	0	0	0
0	3	7	0	5	0	0	0	0	1
0	0	7	0	0	0	2	0	0	0
0	0	7	3	0	2	0	0	1	0

Nedenstående procedure Udfyld løser denne opgave for en værdi af type Papir med dimensioner $n \times n$:

```
Proc TælR[T: Papir] (r: Int, i, j: Int) → (Int)
    if (i<0) ∨ (j<0) ∨ (i ≥ n) ∨ (j ≥ n) ∨ ¬ is(T.(i, j), kryds) → return 0 fi
    if r = 0 → return 1+TælR[T] (0, i+1, j)
    & r = 1 → return 1+TælR[T] (1, i+1, j+1)
    & r = 2 → return 1+TælR[T] (2, i, j+1)
    & r = 3 → return 1+TælR[T] (3, i-1, j+1)
    & r = 4 → return 1+TælR[T] (4, i-1, j)
    & r = 5 → return 1+TælR[T] (5, i-1, j-1)
    & r = 6 → return 1+TælR[T] (6, i, j-1)
    & r = 7 → return 1+TælR[T] (7, i+1, j-1)
    fi
end TælR

Proc TælX[T: Papir] (i, j: Int) → (Int)
    if (i<0) ∨ (j<0) ∨ (i ≥ n) ∨ (j ≥ n) ∨ ¬ is(T.(i, j), kryds) → return 0 fi
    (+ Var x04, x15, x26, x37: Int
        x04 := TælR[T] (0, i, j)+TælR[T] (4, i, j) -1
        x15 := TælR[T] (1, i, j)+TælR[T] (5, i, j) -1
        x26 := TælR[T] (2, i, j)+TælR[T] (6, i, j) -1
        x37 := TælR[T] (3, i, j)+TælR[T] (7, i, j) -1
        return max(x04, x15, x26, x37)
    +)
end TælX

Proc Udfyld[T: Papir, X: Xtabel]
    X := List (List (?-Int | n) | n)
    (+ Var i, j: Int
        i := 0
        do i<n →
            j := 0
            do j<n →
                X.(i, j) := TælX[T] (i, j)
                j := j+1
            od
            i := i+1
        od
    +)
end Udfyld
```

a) Forklar kort, hvorledes dette virker.

b) Angiv den asymptotiske udførelsestid for proceduren Udfyld.

c) Skitsér, hvordan man kan bruge dynamisk programmering til at gøre dette mere effektivt. Angiv den forbedrede udførelsestid. Vink: brug en tabel med dimensioner $(0..8) \times (0..n) \times (0..n)$.