Structuur van Computerprogramma's I: schriftelijke test

Viviane Jonckers

14 januari 2016

Dit examen is **gesloten boek**. Neem voor elke vraag een **volledig nieuw blad** en schrijf op elk blad je naam, richting en het nummer van de vraag, ook als je geen antwoord hebt op die vraag. Antwoorden in potlood worden niet aanvaard!

Veel succes!

1 Lijsten: Inschrijvingen competitie

In het registratieprogramma van een bordspel competitie wordt aan elke deelnemer een uniek inschrijvingsnummer toegekend. Bijgevolg wordt de lijst met alle deelnemers als volgt voorgesteld:

```
(define deelnemers-2016 '((1235 jan) (4566 els) (7536 bart) (7895 ann)))
```

De inschrijving is pas geldig als de deelnemer 12,5 euro heeft gestort. Er wordt aan de deelnemers gevraagd om hun uniek inschrijvingsnummer te vermelden in de mededeling van de overschrijving. Het programma dat instaat voor de boekhouding kan zo gemakkelijk een lijst van alle binnen gekomen stortingen aanmaken. Bijvoorbeeld:

```
(define stortingen-2016 '((7536 12.5) (4566 12.5)))
```

a) Schrijf een procedure `(betaald deelnemers stortingen)` die twee zulke lijsten als input verwacht: een lijst met inschrijvingsnummers en namen en een lijst met inschrijvingsnummers en bedragen. `betaald` geeft een lijst van namen terug waarvoor een storting werd ontvangen. De volgorde is niet belangrijk.

```
> (betaald deelnemers-2016 stortingen-2016)
(els bart)
```

b) Levert je oplossing voor a) een recursief of iteratief proces? Leg in je eigen woorden uit waarom je dit antwoord geeft (argumenteer bondig). Schrijf de andere versie voor a), die *dezelfde* oplossing geeft.

c) Schrijf nu de procedure `(select deelnemers keep? data)` die een veralgemening is van `betaald` uit a), omdat de procedure `keep?` bepaalt of een deelnemer al dan niet in de resultaatlijst voorkomt. `data` is nog steeds een lijst van tupels die *beginnen* met een inschrijvingsnummer, maar de tupels mogen nu ook complexer zijn (d.w.z. meer data bevatten).

d) Maak nu gebruik van de procedure `select` om vraag a) te herimplementeren.

e) Schrijf nu een procedure `(favorieten deelnemers resultaten)` die opnieuw een lijst met deelnemers verwacht en een lijst met resultaten. Eén enkel resultaat is een vier-tupel dat begint met het inschrijvingsnummer, gevolgd door het aantal gewonnen wedstrijden, het aantal gelijke spelen en het aantal verloren wedstrijden. `favorieten` geeft de lijst van favorieten terug, dit zijn spelers die vaker winnen dan verliezen. Schrijf de procedure `favorieten` door gebruik te maken van de procedure `select`.

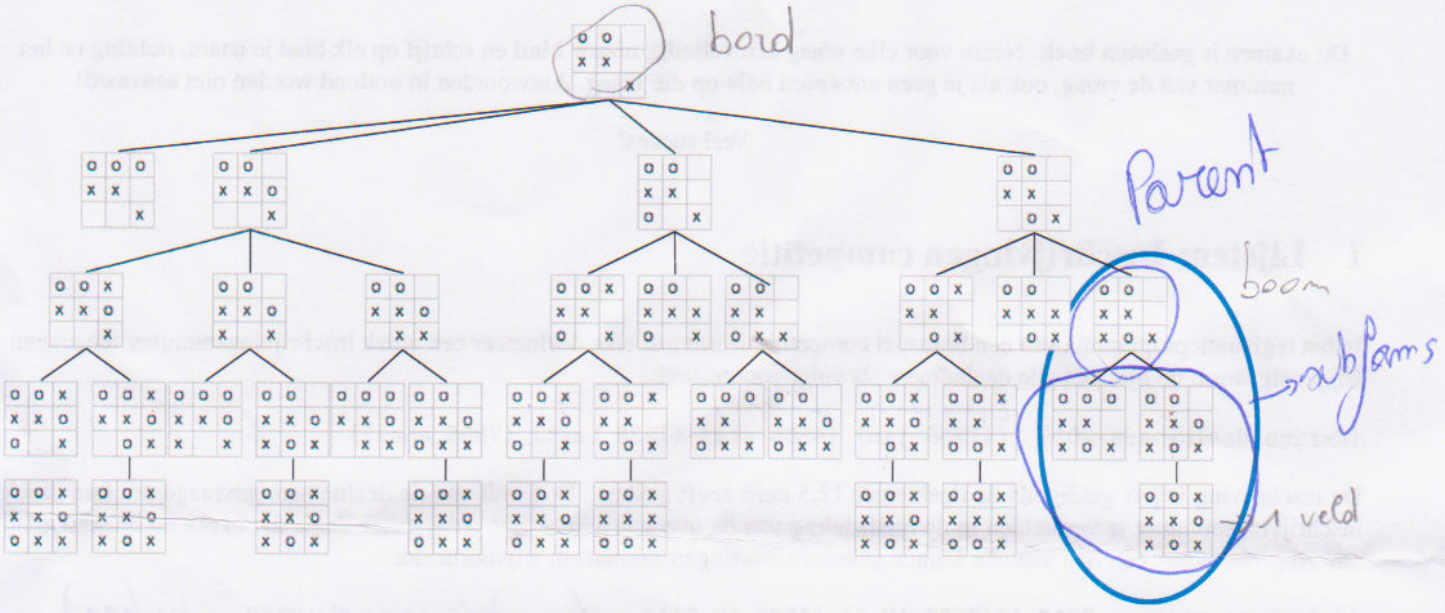
```
(define resultaten-2016 '((1235 15 2 10) (4566 9 2 10) (7536 9 2 10) (7895 15 2 10)))
```

```
> (favoriet deelnemers-2016 resultaten-2016)
(jan ann)
```

2 Bomen: Spelverloop

Kaas-boter-eieren is een spel voor twee spelers, dat gespeeld wordt op een spelbord van 3 op 3 velden. Elke speler heeft een symbool, O of X, dat ze elk om beurt in een leeg veld mogen plaatsen. Wie het eerst drie symbolen op een rij heeft is gewonnen. Als het spelbord vol staat en niemand heeft drie-op-een-rij, dan spreken we van een gelijke stand. In Scheme kunnen we zo een spelbord voorstellen als een getagde lijst van velden. Het bovenste veld in de boom uit de tekening ziet er dan als volgt uit: `'(bord O O #f X X #f #f #f X)`, waarbij `#f` wil zeggen dat het vakje leeg is.

Voor elk spelbord is het mogelijk een boom op te stellen met alle mogelijke spel verlopen. Elke boom is dus een spelbord gevolgd door deelbomen die elk één van de mogelijke nieuwe spelborden voorstelt (elke mogelijke zet van de speler die aan beurt is), dat wil dus zeggen opnieuw een spelbord gevolgd door deelbomen.



Het aangeduide deel van de boom kunnen we in Scheme uitschrijven op de volgende manier:

```
(define deelboom '((bord O O #f X X #f X O X)
                   ((bord O O O X X #f X O X))
                   ((bord O O #f X X O X O X) ((bord O O X X X O X O X)))))
```

a) Voor de komende vragen mag je er van uit gaan dat de boom van velden gegeven is. Schrijf abstracties zodat je in vraag **b** en **c** geen gebruik hoeft te maken van `car` en `cdr` om een boom of bord aan te spreken.

b) Schrijf een procedure `(count a)` die nagaat hoeveel X'en en hoeveel O's er te vinden zijn in een boom van velden. Het antwoord is een cons-cel met in de car het aantal X'en en in de cdr het aantal O's. Zorg er voor dat je procedure slechts 1 maal de boom doorloopt.

```
> (count deelboom)
(17 . 15)
```

c) Merk op dat een boom niet overal even diep is. Dit komt omdat als een speler drie op een rij heeft gemaakt die speler het spel heeft uitgespeeld en er dus geen verdere zetten meer mogelijk zijn. Schrijf een procedure `(min-max boom)` die voor een gegeven boom nagaat wat het minimum aantal stappen is om het spel uit te spelen en wat het maximum aantal stappen is om het spel uit te spelen. Merk op dat je antwoord ook hier een cons-cel zal zijn. Zorg er ook hier voor dat je procedure slechts 1 maal de boom doorloopt.

```
> (min-max deelboom)
(1 . 2)
```

3 Destructieve Operaties: Tombola

Naar jaarlijkse traditie wordt er na de competitie een tombola gehouden. Hierbij trekt een onschuldige kinderhand een aantal bollen met nummers op uit een bokaal. Om dit enigzins te automatiseren beslist de organisatie om dit jaar een Scheme procedure te schrijven die uit een lijst een aantal spelers selecteert aan de hand van de geselecteerde nummers.

Let op: aangezien het gaat om destructieve procedures mag je geen enkele nieuwe cons-cel aanmaken. Je mag hoogstens bestaande cons-cellen hergebruiken, wijzigen of weglaten.

a) Schrijf een procedure `(maak-ring lijst)` die van de deelnemerslijst van de competitie een circulaire lijst maakt.

```
(define deelnemers '((1235 jan)
                     (4566 els)
                     (7536 bart)
                     (7895 ann)))

> (maak-ring deelnemers)

> deelnemers
#0=((1235 jan) (4566 els) (7536 bart) (7895 ann) . #0#)
```

b) Schrijf een procedure `trekking!` die gegeven het resultaat van `maak-ring` en een lijst met getrokken getallen een lijst teruggeeft met de namen van de deelnemers die een prijs hebben gewonnen. Hierbij begin je vanaf het begin van de ring te tellen tot je aan de n-de persoon komt (waarbij n een nummer is uit de trekking). Omdat elke deelnemer maar 1 prijs kan winnen, verwijder je die n-de persoon dan uit de ring. Vervolgens tel je vanaf die positie terug voort met het volgende nummer uit de trekking. Op het einde zijn dus alle personen die een prijs hebben gewonnen verwijderd uit de ring en geeft de procedure een lijstje terug met de personen (inschrijvingsnummer + naam) die een prijs gewonnen hebben.

```
> (trekking! deelnemers '(5 2))
((1235 jan) (7536 bart))
```

4 Objecten: Kerstversiering

a) Ontwerp een ADT `lamp` dat je als object implementeert. Een lamp verandert van toestand elke keer het het bericht `next!` ontvangt. De drie toestanden van een lamp zijn `uit`, `aan` en `knipperend`. Een lamp doorloopt deze toestanden in deze volgorde, d.w.z. een lamp die uit is gaat aan, een lamp die aan is gaat knipperen en een lamp die knippert gaat uit. Het moet ook mogelijk zijn om de huidge toestand van een lamp op te vragen.

b) Ontwerp een ADT `slinger` dat je als object implementeert. Een slinger brengt allemaal *verschillende* lampjes samen die dezelfde toestand hebben. Het aanmaken van een slinger gebeurt op basis van één getal, nl. het aantal lampjes in deze slinger. Net als een lamp, moet een slinger het bericht `next!` verstaan. De slinger zorgt er dan voor dat alle lampjes naar de volgende toestand verspringen. Verder print een slinger zichtzelf schematisch op het scherm als het de boodschap `print` ontvangt. Een slinger van 4 lampjes kan er dus uitzien als één van de volgende drie schema's:

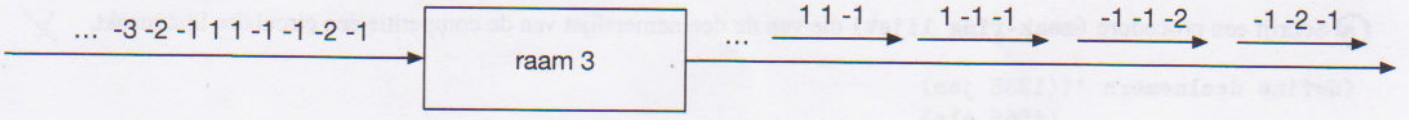
```
-uit-uit-uit-uit-
-aan-aan-aan-aan-
-knipperend-knipperend-knipperend-knipperend-
```

c) Schrijf een kort Scheme programma dat een slinger van vier lampjes aanmaakt, alle lampjes aanzet en de slinger op het scherm toont.

5 Stromen: IJzelvorming

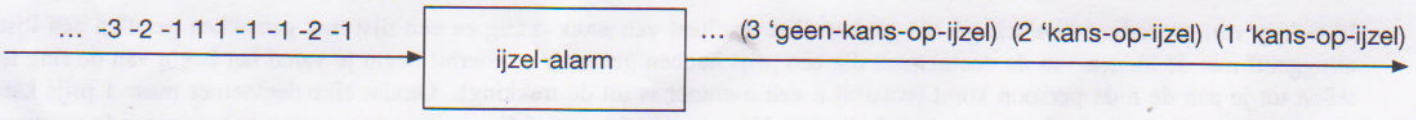
Voor de volgende vragen mag je gebruik maken van de gekende stroom operatoren `cons-stream`, `head`, `tail`, `empty-stream?`, `filter`, `map`, `accumulate` en `append`. `the-empty-stream` kan je gebruiken om een lege stroom voor te stellen.

a) Schrijf een functie `(raam data n)` die als input een stroom `data` neemt en hiervan een stroom van stromen maakt door elke `n` opeenvolgende data-elementen in een stroompje samen te nemen. Veronderstel dat de inputstroom oneindig is.



b) Definieer een stroom die *herhaaldelijk* alle uren bevat (1 tot en met 12). Deze stroom is *oneindig* omdat na 12 uur opnieuw 1 uur volgt.

c) Volgens het K.M.I. is er kans op de vorming van ijzelplekken als de temperatuur 3 uur (of langer) onder het vriespunt ligt (0°C). Schrijf m.b.v. streamoperatoren de procedure `(ijzel-alarm data)` die nagaat voor een gegeven oneindige stroom van metingen of er kans is op ijzelvorming. Het resultaat van `ijzel-alarm` is een stroom van tupels die het uur weergeven en of er al dan niet kans op ijzel is. Bijvoorbeeld zoals op de tekening hieronder.



Teken een diagram voor je oplossing m.b.v. de gebruikte stroom operatoren en geef de Scheme implementatie van `ijzel-alarm`.

loop start stop