Structuur van Computerprogramma's I
Examen Tweede Zit

Woensdag 29 augustus 2018

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. Antwoorden in potlood worden niet aanvaard! Lees elke vraag **volledig** en **aandachtig** voor je aan je oplossing begint! Tevens raden we je aan om zoveel mogelijk gebruik te maken van **abstracties**!

Veel succes!

1 Lijsten

```
(define uitslagen
  '((BEL PAN 3 0)
    (TUN ENG 1 2)
    (BEL TUN 5 2)
    (ENG PAN 6 1)
    (ENG BEL 0 1)
    (PAN TUN 1 2)
    (BEL JAP 3 2)
    (BRA MEX 2 0)))
```

a) Xavier is gepassioneerd door voetbal en statistiekjes. Hij heeft in Scheme, zijn favoriete programmeertaal, een lijst gedefinieerd waarin enkele uitslagen staan van het afgelopen WK. Zo'n uitslag is een lijst met 4 elementen: de thuisploeg, uitploeg, en het aantal punten dat beide ploegen respectievelijk gescoord hebben.

Xavier wil graag dat je hem helpt bij het implementeren van de `totaal-aantal-doelpunten` procedure. Deze procedure neemt twee argumenten: een lijst met uitslagen en een ploeg. De procedure gaat door de lijst en berekent het totaal aantal doelpunten dat gescoord is door de gegeven ploeg.

```
> (totaal-aantal-doelpunten uitslagen 'BEL)
12
```

b) Levert je oplossing voor a) een recursief of iteratief proces op? Leg uit waarom je dit antwoord geeft (argumenteer bondig, en met de juiste terminologie!). Schrijf de andere versie voor a) die *hetzelfde* resultaat teruggeeft.

c) Schrijf nu een procedure `(verwerk-uitslagen lijst proc)`. De procedure `verwerk-uitslagen` overloopt de hele lijst, en past de procedure `proc` toe op elk element uit de lijst en telt dan alle uitkomsten op.

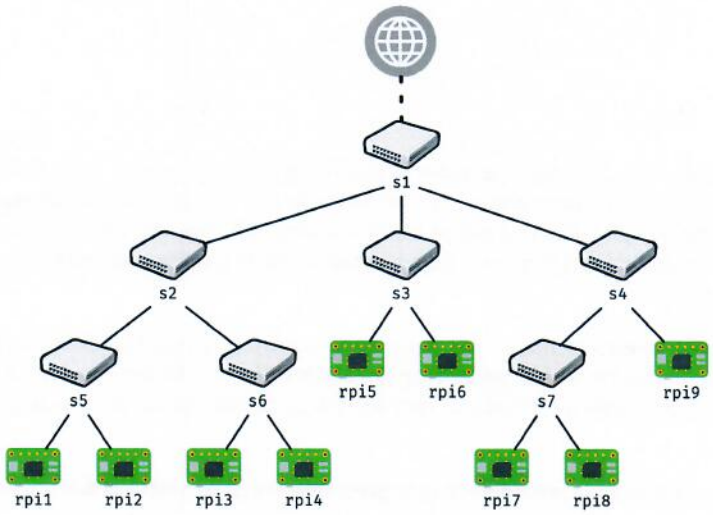
d) Gebruik de procedure `verwerk-uitslagen` om de procedure `totaal-aantal-doelpunten` opnieuw te implementeren.

e) Gebruik de procedure `verwerk-uitslagen` om de procedure `(aantal-gewonnen lijst ploeg)` te implementeren. Deze procedure zal, gegeven een lijst met uitslagen en een ploeg, teruggeven hoeveel matchen deze ploeg gewonnen heeft.

```
> (aantal-gewonnen uitslagen 'BEL)
4
```

2 Bomen

Jan moet voor zijn onderzoek een proefopstelling maken waarbij honderden kleine computertjes verbonden zijn met elkaar en het internet. Omdat de kleine computertjes slechts 1 netwerkaansluiting hebben, moet Jan gebruik maken van meerdere *switches*. Een switch is een speciaal apparaat waarop je computers kan aansluiten, en die het mogelijk maakt om deze computers met elkaar te laten communiceren. Het is zelfs mogelijk twee of meerdere switches met elkaar te verbinden. Hierdoor kunnen alle computers verbonden met individuele switches met elkaar communiceren. Een concreet voorbeeld kan je hieronder zien.



In deze boom kan je zien hoe meerdere kleinere computertjes en switches verbonden zijn met elkaar om zo een netwerk te vormen. Zo kan je zien dat er bovenaan de boom een switch `s1` staat, waarmee enkele andere switches `s2`, `s3`, en `s4` verbonden zijn. Aan deze switches zijn dan weer andere switches of computers verbonden. Op deze manier zijn alle computers in de boom verbonden met elkaar, en is communicatie tussen deze computers en het internet mogelijk.

Deze boom kan als volgt worden voorgesteld in Scheme:

```
> (define netwerk
    '((switch s1) ((switch s2) ((switch s5) ((computer rpi1)) ((computer rpi2)))
                               ((switch s6) ((computer rpi3)) ((computer rpi4))))
                  ((switch s3) ((computer rpi5))
                               ((computer rpi6)))
                  ((switch s4) ((switch s7) ((computer rpi7)) ((computer rpi8)))
                               ((computer rpi9)))))
```

Let op: de boom begint bij de switch `s1`, het internet zit dus niet in deze boom.

a) Schrijf nu een procedure `(via netwerk computer)` die een netwerkboom en het label van een computer als argument heeft. Deze procedure geeft dan een lijst met switches waarmee de computer moet communiceren om het internet te bereiken.

```
> (via netwerk 'rpi7)
(s7 s4 s1)
```

b) Tijdens Jan's experiment produceert elke computer 100 megabit aan data per seconde. Al deze data wordt direct geüpload naar het internet via de switches in het systeem. Zo ontvangt de switch `s7` bijvoorbeeld 200 megabit per seconde aan data van de verbonden computers `rpi7` en `rpi8`. Switch `s4` ontvangt op zijn beurt dan weer 300 megabit per seconde aan data.

Implementeer nu de procedure `(bandbreedte netwerk switch)` die, gegeven een boom en het label van een switch, de totale hoeveelheid data (in megabit per seconde) geeft die deze switch ontvangt.

```
> (bandbreedte netwerk 's4)
300
```

3 Destructieve operatoren

```
(define transacties
  '((boek1 1)
    (boek2 30)
    (boek3 10)
    (boek1 342)
    (boek1 23)
    (boek3 23)
    (boek2 42)))
```

Een winkelketen ontvangt dagelijks een lijst met verkochte artikels van elke vestiging. Om de dagelijkse verkoopscijfers wat overzichtelijker te maken heeft Anne, een medewerkster van de winkelketen, al wat code geschreven om al deze lijsten samen te voegen tot 1 grote lijst. Een voorbeeld van zulke lijst kan je hierboven zien. Elk element van de lijst is een lijst met daarin de naam van het artikel en de verkochte hoeveelheid. Zo kan je uit deze lijst bijvoorbeeld afleiden dat `boek1` respectievelijk 1, 342, en 23 keer verkocht is in 3 verschillende winkels.

Omdat deze lijst toch nog niet heel overzichtelijk is, wil Anne code schrijven die de lijst **destructief** aanpast, zodat de lijst van transacties enkel de totale verkoopscijfers bevat. Elk artikel mag dus slechts 1 keer in de lijst voorkomen, met het totaal aantal keer dat dit artikel verkocht is. Aangezien Anne morgen op vakantie vertrekt, heb jij, de jobstudent, nu de taak gekregen om Anne haar werk af te maken.

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

a) Implementeer de procedure `(voeg-samen! lijst)` die een gegeven lijst van transacties destructief aanpast zoals hierboven beschreven.

```
> (voeg-samen! transacties)
ok
> transacties
((boek1 366) (boek2 72) (boek3 33))
```

4 Objecten

In deze oefening is het de bedoeling om de werking van rederijen, vrachtschepen, en havens simplistisch te simuleren door middel van objecten in Scheme.

Een rederij is een bedrijf dat 1 of meerdere schepen beheert voor bijvoorbeeld vrachtvervoer of toerisme. Elke rederij heeft een "moederhaven", dit is de haven waar het hoofdkantoor van de rederij zich bevindt en waar nieuwe schepen te water worden gelaten. Elke rederij beheert een aantal vrachtschepen, elk met een specifieke capaciteit (uitgedrukt in aantal containers). We gaan er van uit dat, in tegenstelling tot schepen, een haven oneindig groot is en er geen limiet staat op de hoeveelheid containers die een haven kan stockeren.

a) Ontwerp een ADT `schip` dat je als object implementeert. Zoals eerder vermeld heeft elk schip een vaste capaciteit, en begint een schip zijn dienst bij de moederhaven van de rederij die het schip bezit. Eens aangemeerd, kan een schip containers laden en lossen. Bij het laden worden containers van de haven verplaatst naar het schip, bij het lossen worden deze van het schip naar de haven verplaatst. Een schip kan natuurlijk ook varen tussen havens, waarbij het de haven van vertrek verlaat en aanmeert bij de haven van bestemming.

Het laden, lossen, vertrekken, en aanmeren bij havens wordt altijd geinitieerd door een schip. Wees ook zeker dat bij het aanmaken van een schip, het effectief aangemeerd is bij de moederhaven.

Boodscap	**Beschrijving**
`vaar!`	Laat het schip van de huidige haven naar een andere varen
`laad!`	Laadt een gegeven aantal containers op
`los!`	Lost een gegeven aantal containers
`containers`	Geeft terug hoeveel containers zich momenteel op het schip bevinden

b) Ontwerp een ADT `haven` dat je als object implementeert. Een haven heeft een zekere hoeveelheid containers op de kade staan, maar in tegenstelling tot schepen staat er geen limiet op het aantal. Het aantal containers waarmee er gestart wordt, wordt bepaald door de gebruiker.

Boodscap	**Beschrijving**
`vertrek!`	Laat een schip vertrekken
`meer-aan!`	Laat een schip aanmeren
`laad!`	Verwijdert een gegeven aantal containers van de kade
`los!`	Plaatst een gegeven aantal containers op de kade
`containers`	Geeft terug hoeveel containers er momenteel op de kade staan
`schepen`	Geeft een lijst terug van alle aangemeerde schepen

c) Ontwerp een ADT `rederij` dat je als object implementeert. Een rederij heeft een moederhaven en een aantal schepen in bezit. Een rederij kan ook nieuwe schepen aankopen, die dan voor de rederij worden gebouwd en te water gelaten worden aan de moederhaven.

Boodscap	**Beschrijving**
`nieuw-schip!`	Maakt een nieuw schip aan met een gegeven capaciteit
`schepen`	Geeft een lijst terug van alle schepen die de rederij bezit

d) Schrijf nu een scenario uit waarbij je 3 havens maakt voor bijvoorbeeld Amsterdam, Brussel, en Kopenhagen. Tevens maak je voor elke haven een rederij aan. Je laat een van de rederijen een schip aanmaken dat enkele containers oplaadt, naar een andere haven vaart, en daar weer een aantal containers lost.

5 Stromen

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

a) Schrijf een nieuwe stroom operator (`fixed-windows stream n`) die, gegeven een oneindige stroom van elementen, een oneindige stroom van deelstromen teruggeeft. Elke deelstroom is een zogeheten *fixed window*. Dit wil zeggen dat de eerste deelstroom de eerste `n` elementen bevat van de input stroom, de volgende deelstroom bevat dan weer de `n` volgende elementen, enzovoorts.

b) Professor Jonckers is bezorgd. Ze vermoedt dat haar doctoraatsstudenten meer tijd spenderen aan chatten op Slack, dan aan het werken aan hun thesis.

Na wat onderzoek ontdekt ze dat ze via Slack een oneindige stroom, `dagelijkstotaal`, kan binnenhalen die per dag het aantal verzonden berichten weergeeft. Hiermee kan ze aan de slag om statistieken te genereren, en indien nodig, het abonnement op Slack stop te zetten. Concreet zou Professor Jonckers graag weten hoeveel berichten er gemiddeld per week gepost worden op Slack. Kan jij voor haar de stroom operator (`gemiddelde-per-week stroom`) implementeren?

(map -> avg (split ... 7))

↓

((accumulate + 0 ...) (length ...))