Structuur van Computerprogramma's I
Examen Eerste Zit

Donderdag 14 januari 2021

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**.

Merk op dat op dit examen `do`-lussen en named `let`s **niet** gebruikt morgen worden! Verder mag je `set!`, `set-car!` en `set-cdr!` enkel gebruiken als we je vragen om een destructieve procedure te schrijven.

Veel succes!

1 Lijsten en Destructieve Operatoren

In een vakantieplanning app wordt een reisroute opgeslagen als een vlakke lijst. Er zitten mintens 2 steden in de lijst, i.e., het vertrekpunt en de eindbestemming.

(a) Schrijf een functie waarmee je kan controleren of een reisroute een cyclus bevat, i.e., of dezelfde stad 2 keer wordt bezocht.

```
> (cycle? '(brussel parijs nice rome))
#f
> (cycle? '(brussel parijs geneve parijs nice rome))
#t
```

(b) Leg voor elk onderdeel van je oplossing uit of er een recursief of een iteratief proces wordt gegenereerd.

(c) Schrijf een destructieve procedure waarmee een kort rijtje van nieuwe steden aan een reisroute kan toegevoegd worden. Naast de lijst van nieuw toe te voegen steden wordt ook de stad waarachter deze nieuwe steden moeten komen als argument meegegeven. Let op! Hier mag je geen nieuwe cons-cellen aanmaken.

```
> (define my-route '(brussel parijs nice rome))
> (insert! my-route '(st-marie cannes) 'parijs)
> my-route
(brussel parijs st-marie cannes nice rome)
```

2 Bomen

Een Lambda-notebook is een speciaal formaat om cursusnota's over Scheme op te maken. Een notebook bestaat uit een groep cellen. Er zijn twee soorten basiscellen: *codecellen* en *tekstcellen*. Cellen kunnen willekeurig worden gegroepeerd in een *groepcel* waardoor een geneste structuur ontstaat.

Onderstaande figuur is een voorbeeld. Naast het voorbeeld staat een Scheme lijst die hetzelfde voorbeeld voorstelt. Let op, wegens plaatsgebrek is het voorbeeld heel klein. In het algemeen bevat een Lambda-notebook veel meer cellen en zijn er ook meer levels van nesting.

```
(define les1
  '(*group*
     (*text* "Deze les gaat over ...")
     (*code* (define (fac ...)))
     (*group*
      (*text* "Je kan een ...")
      (*code* (define (make-trace ....)))
      (*code* (define traced-fac ...))
      (*text* "Als je dit wil ...")
      (*code* (traced-fac 5)))
     (*text* "In conclusie: ...")))
```

(a) In de representatie van het voorbeeld gebruiken we een techniek van manifeste typering. Waar? En waarom is dat nuttig?

(b) Schrijf een functie `tel-cellen` die telt hoeveel tekstcellen en hoeveel codecellen er in een gegeven Lambda-notebook zitten. Het is hier de bedoeling dat je procedure maar 1 keer door de boom loopt. In het voorbeeld zitten 4 tekstcellen en 4 codecellen.

```
> (tel-cellen les1)
(4 . 4)
```

(c) Een Lambda-notebook is *didactisch-ok* als er in elke groep juist evenveel tekstcellen als codecellen zitten en de geneste groepen op hun beurt ook didactisch-ok zijn. Schrijf een functie die controleert of een gegeven Lambda-notebook didactisch-ok is. Het voorbeeld is niet didactisch-ok want in de (enige) geneste groep zitten 3 codecellen en maar 2 tekstcellen en in de top-groep zitten dan weer meer tekstcellen dan codecellen.

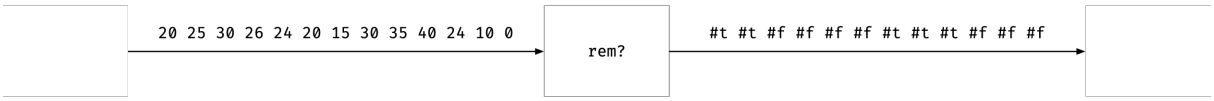
```
> (didactisch-ok? les1)
#f
```

3 Streams

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

In een moderne auto wordt constant de buitentemperatuur gemonitord en natuurlijk is ook de snelheid op elk moment gekend. Veronderstel een controller die elke de huidige seconde snelheid en temperatuur op een stroom zet.

(a) Implementeer een procedure `rem?` die gegeven een stroom van snelheden een stroom genereert met `#t` of `#f` naargelang de auto afremt of niet. Een auto is aan het afremmen als de snelheid van de vorige meting strikt groter is dan de huidige snelheid.



(b) Implementeer als onderdeel van een veiligheidssysteem een procedure `check` die een signaal `'aan` of `'uit` geeft aan een rembekrachtigingssysteem telkens de buitentemperatuur onder het vriespunt zit en de wagen op hetzelfde moment aan het afremmen is. Maak eerst een schets van je oplossing.

Let op! Je moet de oplossing formuleren als een combinatie van de gekende procedures `map-stream`, `stream-filter`, `accumulate`, enz, en eventueel je nieuwe procedure `rem?`.

4 Theorie

In de vakantieplanning app uit vraag 1 kan je de kostprijs van je reisroute bereken. Er zijn verschillende opties voor brandstoftype, brandstofprijs, gemiddeld verbruik, kortste of snelste route, ect. Jan bestudeert de code en vindt volgende functie.

```
(define (calculate-cost route distance price)
  (if (null? (cdr route))
      0
      (+ (price (distance (car route) (cadr route)))
         (calculate-cost (cdr route) distance price))))
```

(a) Is de functie `calculate-cost` een hogere orde functie? Zo ja, waarom? Zo neen, waarom niet?

(b) Wat verderop vindt Jan ook een definitie voor een *segment* object.

```
(define (make-segment a b)
  (define (begin) a)
  (define (end) b)
  (define (consecutive other)
    (eq? b (other 'begin)))
  (define (dispatch m)
    (cond
      ((eq? m 'begin) (begin))
      ((eq? m 'end) (end))
      ((eq? m 'consecutive) consecutive)
      (else 'error)))
  dispatch)
```

Teken zorgvuldig in een omgevingsmodel uit wat het resultaat is van de evaluatie van volgende uitdrukkingen

```
(define s1 (make-segment 'brussel 'parijs))
(define s2 (make-segment 'parijs 'nice))
((s1 'consecutive) s2)
```

(c) Je kan objecten maken zoals we dat in de cursus hebben gezien in elke programmeertaal die volgende eigenschappen heeft:

1. Geneste functies/procedures ondersteunen met lexicale scoping regels
2. Functies/procedures als eerste klas objecten ondersteunen

Leg in je eigen woorden uit waarom beide eigenschappen nodig zijn. Gebruik daarbij het voorbeeld van het segment object om je uitleg concreet te maken.