

Voorafgaandelijk: de vragen zijn geformuleerd in functie van van de gepubliceerde nota's; deze mogen, samen met persoonlijke nota's en het referentieboek van Abelson & Sussman, geraadpleegd worden tijdens de test. Het gebruik van (electro-)mechanische hulpmiddelen is niet toegestaan. Gelieve elk antwoord te beantwoorden op een apart blad en te beperken tot maximum één blad. Geef voor alle vragen één blad af en zet bovenaan elk blad je naam en het nummer van de vraag die beantwoord wordt.

*Succes!
Theo D'Hondt
14 juni 2007*

Vraag #1: beschouw deel 1a (De meta-circulaire evaluator).

Gevraagd wordt een uitbreiding te maken van mc-eval met een nieuwe special form:

```
(for-list <var> <list> <body>)
```

met <var> een variabele, <list> een (niet-geneste) lijst, en <body> een Scheme-expressie. Voorbeeld van gebruik is:

```
(for-list i '(1 2) (display i))  
> "1"  
> "2"  
> 2
```

De semantiek van de for-list volgt deze afgeleide expressie:

```
(let (rest-l <list>))  
  (define (iter)  
    (if (null? rest-l)  
        the-unspecified-value  
        (begin  
          (set! <var> (car rest-l))  
          <body>  
          (set! rest-l (cdr rest-l))  
          (iter))))  
  (iter))
```

Vraag #2: beschouw deel 1d (Logisch programmeren).

We beschouwen de volgende feitenbank die weergeeft tot welk departement personeelsleden behoren. Een personeelslid kan tot meerdere departementen behoren:

```
(assert! (lid-van wolfgang dinf))
(assert! (lid-van wolfgang we))
(assert! (lid-van viviane dinf))
(assert! (lid-van wim we))
(assert! (lid-van rosette ozr))
```

Daarnaast stellen we een feitenbank op die de tijdstippen bevat waarop vergaderingen van een departement plaatsvinden:

```
(assert! (meeting dinf (dinsdag 13)))
(assert! (meeting we (donderdag 14)))
(assert! (meeting ozr (woensdag 10)))
(assert! (meeting we (vrijdag 9)))
(assert! (meeting alle (dinsdag 13)))
```

a) Schrijf een regel die een overzicht geeft van alle departementen ?dep die op dag ?dag een vergadering hebben.

b) Schrijf een regel die toont op welke dagen ?dag een bepaalde persoon ?p vergaderingen heeft in een bepaald departement ?dep. Daarbij geldt dat vergaderingen met departementsnaam "alle", door alle personeelsleden gevolgd moeten worden ongeacht het (de) departement(en) waartoe ze behoren.

Bijvoorbeeld:

```
(meeting-per-persoon wolfgang ?dep ?dag ?tijd)
> (meeting-per-persoon wolfgang dinf dinsdag 13)
> (meeting-per-persoon wolfgang we donderdag 14)
> (meeting-per-persoon wolfgang we vrijdag 9)
> (meeting-per-persoon wolfgang alle dinsdag 13)
```

c) Schrijf een regel die een lijst weergeeft van personeelsleden ?p die overlappende meetings hebben.

Vraag #3: beschouw deel 2c (Registermachines).

Herschrijf onderstaande Scheme functie in registermachinecode in recursieve stijl. Geef aan hoe deze functie vanuit een omliggend registermachineprogramma moet aangeroepen worden

```
(define (delete x l)
  (cond
    ((null? l) l)
    ((eq? x (car l)) (delete x (cdr l)))
    (else (cons (car l) (delete x (cdr l))))))
```

Vraag #4: beschouw deel 2d (Garbage collection).

Veronderstel dat voor een bepaalde toepassing van het bewuste stop-and-copy geheugenbeheersysteem er een adres <barrier> kan gevonden worden waarvoor geldt dat:

- alle cons-cellen met een adres kleiner dan <barrier> bewaren hun adres tijdens een garbage collection,
- indien de car en cdr velden van een dergelijke cons-cel een adres bevat, dan is dit adres steeds kleiner dan <barrier>.

Met andere woorden, het stuk van het geheugen beneden <barrier> is volledig invariant voor garbage collection.

Pas de code van de bestaande stop-and-copy garbage collector aan om van dit gegeven gebruik te maken ten einde overbodige bewerkingen te elimineren.

Vraag #5: beschouw deel 2e (Vertaling).

Beschouw de volgende registermachinecode die het resultaat is van de vertaling van een Scheme expressie:

```
((env continue)
 (proc argl val)
 ((save continue)
  (save env)
  (assign proc (op lookup-variable-value) (const p) (reg env))
  (assign argl (const ()))
  (test (op primitive-procedure?) (reg proc))
  (branch (label label-2))
label-1
  (assign continue (label label-3))
  (assign val (op compiled-procedure-entry) (reg proc))
  (goto (reg val))
label-2
  (assign val (op apply-primitive-procedure) (reg proc) (reg argl))
label-3
  (restore env)
  (restore continue)
  (test (op false?) (reg val))
  (branch (label label-8))
label-4
  (assign val (op lookup-variable-value) (const q) (reg env))
  (test (op false?) (reg val))
  (branch (label label-6)))
```

```
label-5  
  (assign val (const 0))  
  (goto (reg continue))  
label-6  
  (assign val (const 1))  
  (goto (reg continue))  
label-7  
label-8  
  (assign val (const 2))  
  (goto (reg continue))  
label-9))
```

Merk op dat de labels "anoniem" werden gemaakt. Gevraagd wordt de parameters te reconstrueren van de compile aanroep die deze code heeft gegenereerd.