



Problema 14

28 Marzo 2012

Descrizione

Riformula e sperimenta in Java ciascuna delle seguenti procedure Scheme discusse a lezione:

1.

```
(define gcd
  (lambda (x y)
    (cond ((= x y) x)
          ((< x y) (gcd x (- y x)))
          (else (gcd (- x y) y))
          )))
```

 ; valore: intero
; x, y > 0 interi
2.

```
(define mul
  (lambda (m n)
    (cond ((= n 0) 0)
          ((even? n) (mul (* 2 m) (quotient n 2)))
          (else (+ (mul (* 2 m) (quotient n 2)) m))
          )))
```

 ; valore: intero
; m, n: interi non negativi
3.

```
(define length ...)
```

 ; lunghezza di una lista
4.

```
(define reverse ...)
```

 ; rovesciamento di una lista
5.

```
(define append ...)
```

 ; fusione di due liste

Per verificare se un intero è pari — (`even? n`) — è sufficiente confrontare con zero il resto della divisione per due, in Java: `n % 2 == 0`.

Per quanto riguarda le procedure che operano sulle liste, considera esclusivamente *liste di interi*. A tale proposito utilizza la classe (atipica) `IntList`, che emula in Java le liste di Scheme, purché gli elementi siano tutti numeri interi. Il protocollo per operare con questo tipo di liste è riformulato come segue:

Tipo `IntList`:

<code>IntList.NULL</code>	costante lista vuota [<code>null</code>]
<code>IntList.listNull(lst)</code>	predicato per verificare se una lista è vuota [<code>null?</code>]
<code>IntList.listCar(lst)</code>	primo elemento di una lista [<code>car</code>]
<code>IntList.listCdr(lst)</code>	lista che si ottiene eliminando il primo elemento [<code>cdr</code>]
<code>IntList.listCons(n, lst)</code>	lista che si ottiene aggiungendo un nuovo elemento all'inizio [<code>cons</code>]

Nota che in Java occorre premettere ai nomi di “costanti” e “procedure” il nome della classe che li definisce. Per esempio, la lista `'(1 2 3)` può essere definita come segue:

```
IntList.listCons(1,IntList.listCons(2,IntList.listCons(3,IntList.NULL)))
```