Laboratorio di Programmazione a.a. 2012-13

Problema 14

3 Aprile 2013

Descrizione

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

```
1.      (define gcd                ; valore: intero
          (lambda (x y)            ; x, y > 0 interi
            (cond ((= x y) x)
                  ((< x y) (gcd x (- y x)))
                  (else    (gcd (- x y) y))
                  )))

2.      (define mul                ; valore: intero
          (lambda (m n)            ; m, n: interi non negativi
            (cond ((= n 0)   0)
                  ((even? n)    (mul (* 2 m) (quotient n 2))   )
                  (else      (+ (mul (* 2 m) (quotient n 2)) m))
                  )))

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`:

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

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)))
```