Corso di Programmazione

II Accertamento del 24 Marzo 2003 / A

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte, utilizzando se necessario lo spazio bianco disponibile.

1. Procedure in Scheme

In relazione alla procedura *h*, calcola i risultati della valutazione delle espressioni Scheme:

(h + '(1)) *(h + '(1 3))*

(h + '(1 3 5)) *(h + '(1 3 5 7 9))*

e ipotizza il risultato della generica valutazione *(h + '(1 3 5 ... 2n–1))* .

```
(define h
  (lambda (f x)
    (cond
     ((null? x) null)
     ((null? (cdr x)) x)
     (else
      (cons (car x)
            (h f (cons (f (car x) (cadr x))
                       (cddr x)) )) )
     )))
```

2. Procedure in Scheme

Completa la procedura *filtro* in Scheme che, dato un predicato (procedura a valori booleani) *incluso?* e una lista *fonte*, assume come valore la lista degli elementi di *fonte* per i quali il predicato *incluso?* è vero. Informalmente, *filtro* toglie da *fonte* gli elementi *x* tali che *(incluso? x)* è falso.

```
(define filtro
  (lambda (incluso? fonte)
    (cond ( __________
            __________ )
          ((incluso?  __________ )
           ( __________
             (filtro incluso? (cdr fonte)) ) )
          (else
            ____________________________ ) )
    ))
```

3. Definizione di procedure in Scheme

Definisci una procedura a valori procedurali *multiplo-di?*, tale che *(multiplo-di? x)* sia la procedura che verifica se un numero naturale è un multiplo di *x*. Per esempio, l'espressione *((multiplo-di? 2) n)* assume il valore vero se e solo se *n* è pari; altrimenti il valore è falso.

4. Definizione di procedure in Scheme

Utilizzando la procedura *filtro* dell'esercizio 2 e la procedura *multiplo-di?* dell'esercizio 3, definisci una procedura *selezione* che, date due liste *x* e *y* di naturali positivi, assuma come valore la lista di tutti e soli gli elementi di *y* che non sono multipli di elementi di *x*. Per esempio:

(selezione '(3 5) '(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20))

$\rightarrow$ *(1 2 4 7 8 11 13 14 16 17 19)*

(Non sono ammesse soluzioni che non facciano un opportuno uso di *filtro* e *multiplo-di?*.)

5. Realizzazione di strutture dati

Considera la seguente versione del gioco del Nim: all'inizio ci sono *n* scatole, contenenti rispettivamente *n* monete, *n–1* monete, ..., *2* monete, *1* moneta. I due giocatori, a turno, possono prelevare un certo numero di monete (delle monete rimaste) da una sola delle *n* scatole. Perde chi preleva l'ultima moneta.

Progetta una struttura dati basata sulle liste e definisci in Scheme le procedure previste dal protocollo qui di seguito indicato, che comprende un "costruttore" con parametro *n* per creare la configurazione iniziale, un'operazione per realizzare la mossa di un giocatore che intende prelevare dall'*i*-ima scatola *k* monete (se la mossa è lecita) e una predicato per verificare se il giocatore di turno ha perso. Le procedure devono essere puramente funzionali.

```
;  Struttura dati "Gioco del Nim"

;  (configurazione-iniziale <num>)    :  numero -> gioco
;  (preleva <scatola> <num> <gioco>)  :  numero x numero x gioco -> gioco
;  (perso? <gioco>)                   :  gioco -> booleano
```

Corso di Programmazione

II Accertamento del 24 Marzo 2003 / B

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte, utilizzando se necessario lo spazio bianco disponibile.

1. Procedure in Scheme

In relazione alla procedura *h*, calcola i risultati della valutazione delle espressioni Scheme:

*(h * '(1))* *(h * '(1 2))*

*(h * '(1 2 3))* *(h * '(1 2 3 4 5))*

e ipotizza il risultato della generica valutazione *(h * '(1 2 3 ... n))* .

```
(define h
  (lambda (f x)
    (cond
     ((null? x) null)
     ((null? (cdr x)) x)
     (else
      (cons (car x)
            (h f (cons (f (car x) (cadr x))
                       (cddr x)) )) )
     )))
```

2. Procedure in Scheme

Completa la procedura *filtro* in Scheme che, dato un predicato (procedura a valori booleani) *escluso?* e una lista *fonte*, assume come valore la lista degli elementi di *fonte* per i quali il predicato *escluso?* è falso. Informalmente, *filtro* toglie da *fonte* gli elementi *x* tali che *(escluso? x)* è vero.

```
(define filtro
  (lambda (escluso? fonte)
    (cond ( __________
            __________ )
          ((escluso?  __________ )
            ____________________________ )
          (else
           ( __________
             (filtro escluso? (cdr fonte)) ) ))
    ))
```

3. Definizione di procedure in Scheme

Definisci una procedura a valori procedurali *divisore-di?*, tale che *(divisore-di? x)* sia la procedura che verifica se un naturale positivo è un divisore di *x*. Per esempio, l'espressione *((divisore-di? 64) n)* assume valore vero se e solo se $n \in \{1, 2, 4, 8, 16, 32, 64\}$; altrimenti è falsa.

4. Definizione di procedure in Scheme

Utilizzando la procedura *filtro* dell'esercizio 2 e la procedura *divisore-di?* dell'esercizio 3, definisci una procedura *selezione* che, date due liste *x* e *y* di naturali positivi, assuma come valore la lista di tutti e soli gli elementi di *y* che dividono ogni elemento di *x*. Per esempio:

(selezione '(900 1500) '(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20))

$\rightarrow$ *(1 2 3 4 5 6 10 12 15 20)*

(Non sono ammesse soluzioni che non facciano un opportuno uso di *filtro* e *divisore-di?*.)

5. Realizzazione di strutture dati

Considera la seguente versione del gioco del Nim: all'inizio ci sono *n* scatole, contenenti rispettivamente *1* moneta, *3* monete, ..., *2n–3* monete, *2n–1* monete. I due giocatori, a turno, possono prelevare un certo numero di monete (delle monete rimaste) da una sola delle *n* scatole. Perde chi preleva l'ultima moneta.

Progetta una struttura dati basata sulle liste e definisci in Scheme le procedure previste dal protocollo qui di seguito indicato, che comprende un "costruttore" con parametro *n* per creare la configurazione iniziale, un'operazione per realizzare la mossa di un giocatore che intende prelevare dall'*i*-ima scatola *k* monete (se la mossa è lecita) e una predicato per verificare se il giocatore di turno ha perso. Le procedure devono essere puramente funzionali.

```
;  Struttura dati "Gioco del Nim"

;  (configurazione-iniziale <num>)    :  numero -> gioco
;  (preleva <scatola> <num> <gioco>)  :  numero x numero x gioco -> gioco
;  (perso? <gioco>)                   :  gioco -> booleano
```

Corso di Programmazione

II Accertamento del 24 Marzo 2003 / A

cognome e nome

1. Procedure in Scheme

(h + '(1)) → (1) *(h + '(1 3)) → (1 4)*

(h + '(1 3 5)) → (1 4 9)

(h + '(1 3 5 7 9)) → (1 4 9 16 25)

caso generale:

(h + '(1 3 5 ... 2n–1)) →
→ (1 4 9 ... n^2)

2. Procedure in Scheme

Definizione completa:

```
(define filtro
  (lambda (incluso? fonte)
    (cond ( (null? fonte)
            null )
          ((incluso? (car fonte) )
           ( (cons (car fonte) (filtro incluso? (cdr fonte)) ) )
          (else
            (filtro incluso? (cdr fonte))
           ))
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define multiplo-di?
  (lambda (x)
    (lambda (y) (= (remainder y x) 0))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y)
    (if (null? x)
        y
        (selezione
         (cdr x)
         (filtro
          (lambda (z) (not ((multiplo-di? (car x)) z)))
          y)
        )
    )))
```

5. Realizzazione di strutture dati

Definizione:

```
(define configurazione-iniziale
  (lambda (num)
    (if (= num 1)
        '(1)
        (cons num (configurazione-iniziale (- num 1)))
    )))

(define preleva
  (lambda (scatola num gioco)  ; prelievo valido
    (if (= scatola 1)
        (cons (- (car gioco) num) (cdr gioco))
        (cons (car gioco)
              (preleva (- scatola 1) num (cdr gioco)))
    )))

(define perso?
  (lambda (gioco)
    (= (somma gioco) 1)
    ))

(define somma
  (lambda (gioco)
    (if (null? gioco)
        0
        (+ (car gioco) (somma (cdr gioco)))
    )))
```

Corso di Programmazione

II Accertamento del 24 Marzo 2003 / B

cognome e nome

1. Procedure in Scheme

*(h * '(1)) → (1)* *(h * '(1 2)) → (1 2)*

*(h * '(1 2 3)) → (1 2 6)*

*(h * '(1 2 3 4 5)) → (1 2 6 24 120)*

caso generale:

*(h * '(1 2 3 ... n)) →*
→ (1 2 6 ... n!)

2. Procedure in Scheme

Definizione completa:

```
(define filtro
  (lambda (escluso? fonte)
    (cond ( (null? fonte)
            null )
          ((escluso? (car fonte) )
            (filtro escluso? (cdr fonte)) )
          (else
           ( (cons (car fonte) (filtro escluso? (cdr fonte)) )
           ))
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define divisore-di?
  (lambda (x)
    (lambda (y) (= (remainder x y) 0))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y)
    (if (null? x)
        y
        (selezione
         (cdr x)
         (filtro
          (lambda (z) (not ((divisore-di? (car x)) z)))
          y)
        )
    )))
```

5. Realizzazione di strutture dati

Definizione:

```
(define configurazione-iniziale
  (lambda (num)
    (if (= num 1)
        '(1)
        (append (configurazione-iniziale (- num 1))
                (list (- (* 2 num) 1)))
    )))

(define preleva
  (lambda (scatola num gioco)  ; prelievo valido
    (if (= scatola 1)
        (cons (- (car gioco) num) (cdr gioco))
        (cons (car gioco)
              (preleva (- scatola 1) num (cdr gioco)))
    )))

(define perso?
  (lambda (gioco)
    (= (somma gioco) 1)
    ))

(define somma
  (lambda (gioco)
    (if (null? gioco)
        0
        (+ (car gioco) (somma (cdr gioco)))
    )))
```