Corso di Programmazione

II Accertamento del 15 Marzo 2002 / A

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando i fogli protocollo.

1. Procedure in Scheme

Cosa calcola la procedura *f*?
Calcola i risultati della valutazione delle espressioni Scheme:

(f '(a)) (f '(a b)) (f '(a b c)) (f '(a b c d))

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

```
(define f
  (lambda (x)
    (if (null? x) null
          (cons (car x) (g (cdr x))) )))

(define g
  (lambda (x)
    (if (null? x) null
         (f (cdr x)) )))
```

2. Realizzazione di strutture dati

Descrivi sinteticamente una possibile rappresentazione in Scheme delle matrici rettangolari (numeriche). Quindi definisci le procedure *matrice*, per costruire una matrice rettangolare di elementi nulli, ed *elemento* per acquisire l'elemento corrispondente a una coppia di indici.

```
;  Operazioni relative alle matrici
;  (matrice <righe> <colonne>)  :  naturale x naturale -> matrice
;  (elemento <matrice> <indice> <indice>)
                    :  matrice x naturale x naturale -> numero
   ...   ...
```

3. Applicazione di strutture dati

Supponi che sia data un'opportuna realizzazione degli alberi binari, accessibile attraverso le operazioni introdotte qui di seguito. Utilizzando tali operazioni definisci una procedura in Scheme per verificare se un albero binario non vuoto è uno *heap*, cioè se il valore numerico di ogni nodo è minore del valore associato al corrispondente nodo padre.

```
;  Operazioni relative agli alberi binari
;  (singoletto? <albero>)           :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> numero
```

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dati due naturali *x*, *y* e una lista *L* di numeri naturali, calcolare la lista di tutti e soli gli elementi di *L* compresi nell'intervallo *[x, y]*.

5. Dimostrazioni per induzione

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)*, dove *L* è una lista non vuota di numeri, è dato dal minimo elemento di *L*.

In particolare:

```
(define f
  (lambda (L) (f-iter (cdr L) (car L)) ))

(define f-iter
  (lambda (L x)
    (cond ((null? L) x)
          ((< (car L) x)
           (f-iter (cdr L) (car L)))
          (else (f-iter (cdr L) x)) )))
```

- Scrivi formalmente la proprietà che intendi dimostrare per induzione.
- Scrivi formalmente la proprietà che esprime il caso base.
- Scrivi formalmente l'ipotesi induttiva.
- Scrivi formalmente la proprietà che si deve dimostrare come passo induttivo.
- Dimostra formalmente il caso base.
- Dimostra formalmente il passo induttivo.

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / B

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando i fogli protocollo.

1. Procedure in Scheme

Cosa calcola la procedura *f*?
Calcola i risultati della valutazione delle espressioni Scheme:

(f '(a)) (f '(a b)) (f '(a b c)) (f '(a b c d))

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

```
(define f
  (lambda (x)
    (if (null? x) null
        (g (cdr x)) )))

(define g
  (lambda (x)
    (if (null? x) null
        (cons (car x) (f (cdr x))) )))
```

2. Realizzazione di strutture dati

Descrivi sinteticamente una possibile rappresentazione in Scheme delle matrici rettangolari (numeriche). Quindi definisci le procedure *matrice*, per costruire una matrice rettangolare di elementi nulli, ed *assegnazione* per assegnare l'elemento corrispondente a una coppia di indici.

```
;  Operazioni relative alle matrici
;  (matrice <righe> <colonne>)  :  naturale x naturale -> matrice
;  (assegnazione <matrice> <indice> <indice> <valore>)
                  :  matrice x naturale x naturale x numero -> matrice
   ...   ...
```

3. Applicazione di strutture dati

Supponi che sia data un'opportuna realizzazione degli alberi binari, accessibile attraverso le operazioni introdotte qui di seguito. Utilizzando tali operazioni definisci una procedura in Scheme per verificare se un albero binario non vuoto è uno *heap*, cioè se il valore numerico di ogni nodo è maggiore del valore associato al corrispondente nodo padre.

```
;  Operazioni relative agli alberi binari
;  (singoletto? <albero>)           :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> numero
```

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dati due naturali *x*, *y* e una lista *L* di numeri naturali, calcolare la lista di tutti e soli gli elementi di *L* che non appartengono all'intervallo *[x, y]*.

5. Dimostrazioni per induzione

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)*, dove *L* è una lista non vuota di numeri naturali positivi, è dato dal massimo comun divisore (*gcd*) degli elementi di *L*. In particolare:

```
(define f
  (lambda (L) (f-iter (cdr L) (car L)) ))

(define f-iter
  (lambda (L x)
    (if (null? L) x
        (f-iter (cdr L) (gcd (car L) x)))
  ))
```

- Scrivi formalmente la proprietà che intendi dimostrare per induzione.
- Scrivi formalmente la proprietà che esprime il caso base.
- Scrivi formalmente l'ipotesi induttiva.
- Scrivi formalmente la proprietà che si deve dimostrare come passo induttivo.
- Dimostra formalmente il caso base.
- Dimostra formalmente il passo induttivo.

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / C

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando i fogli protocollo.

1. Procedure in Scheme

Cosa calcola la procedura *f*?
Calcola i risultati della valutazione delle espressioni Scheme:

(f '(a)) (f '(a b)) (f '(a b c)) (f '(a b c d))

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

```
(define f (lambda (x) (g x null) ))

(define g
  (lambda (x y)
    (cond ((null? x) y)
          ((null? (cdr x)) y)
          (else (g (cddr x)
                   (cons (cadr x) y))) )))
```

2. Realizzazione di strutture dati

Descrivi sinteticamente una possibile rappresentazione in Scheme delle matrici rettangolari (numeriche). Quindi definisci le procedure *numero-righe*, *numero-colonne*, per determinare le dimensioni, ed *elemento* per acquisire l'elemento corrispondente a una coppia di indici.

```
;  Operazioni relative alle matrici
;  (numero-righe <matrice>)     :  matrice -> naturale
;  (numero-colonne <matrice>)   :  matrice -> naturale
;  (elemento <matrice> <indice> <indice>)
                   :  matrice x naturale x naturale -> numero
   ...   ...
```

3. Applicazione di strutture dati

Supponi che sia data un'opportuna realizzazione degli alberi binari, accessibile attraverso le operazioni introdotte qui di seguito. Utilizzando tali operazioni definisci una procedura in Scheme per verificare se c'è un nodo di valore dato in un *albero binario di ricerca* non vuoto. Un albero binario di ricerca è tale che il sottoalbero sinistro di un nodo di valore *n* contiene solo nodi di valore minore di *n* e quello destro solo nodi di valore maggiore di *n*.

```
;  Operazioni relative agli alberi binari
;  (singoletto? <albero>)           :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> numero
```

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dati due naturali *x*, *y* e una lista *L* di numeri naturali, calcolare la lista di tutti e soli gli elementi di *L* che sono contemporaneamente multipli di *x* e di *y*.

5. Dimostrazioni per induzione

Considera la procedura *g* e dimostra per induzione che il valore dell'espressione *(g L)*, dove *L* è una lista non vuota di numeri, è dato dal massimo elemento di *L*.

In particolare:

```
(define g
  (lambda (L) (g-iter (cdr L) (car L)) ))

(define g-iter
  (lambda (L x)
    (cond ((null? L) x)
          ((< (car L) x) (g-iter (cdr L) x))
          (else
           (g-iter (cdr L) (car L))) )))
```

- Scrivi formalmente la proprietà che intendi dimostrare per induzione.
- Scrivi formalmente la proprietà che esprime il caso base.
- Scrivi formalmente l'ipotesi induttiva.
- Scrivi formalmente la proprietà che si deve dimostrare come passo induttivo.
- Dimostra formalmente il caso base.
- Dimostra formalmente il passo induttivo.

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / D

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando i fogli protocollo.

1. Procedure in Scheme

Cosa calcola la procedura *f*?
Calcola i risultati della valutazione delle espressioni Scheme:

(f '(a)) (f '(a b)) (f '(a b c)) (f '(a b c d))

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

```
(define f (lambda (x) (g x null) ))

(define g
  (lambda (x y)
    (cond ((null? x) y)
          ((null? (cdr x)) (cons (car x) y))
          (else (g (cddr x)
                   (cons (car x) y))) )))
```

2. Realizzazione di strutture dati

Descrivi sinteticamente una possibile rappresentazione in Scheme delle matrici rettangolari (numeriche). Quindi definisci le procedure *numero-righe*, *numero-colonne*, per determinare le dimensioni, ed *assegnazione* per assegnare l'elemento corrispondente a una coppia di indici.

```
;  Operazioni relative alle matrici
;  (numero-righe <matrice>)     :  matrice -> naturale
;  (numero-colonne <matrice>)   :  matrice -> naturale
;  (assegnazione <matrice> <indice> <indice> <valore>)
                   :  matrice x naturale x naturale x numero -> matrice
   ...   ...
```

3. Applicazione di strutture dati

Supponi che sia data un'opportuna realizzazione degli alberi binari, accessibile attraverso le operazioni introdotte qui di seguito. Utilizzando tali operazioni definisci una procedura in Scheme per verificare se c'è un nodo di valore dato in un *albero binario di ricerca* non vuoto. Un albero binario di ricerca è tale che il sottoalbero sinistro di un nodo di valore *n* contiene solo nodi di valore maggiore di *n* e quello destro solo nodi di valore minore di *n*.

```
;  Operazioni relative agli alberi binari
;  (singoletto? <albero>)           :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> numero
```

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dati due naturali *x*, *y* e una lista *L* di numeri naturali, calcolare la lista di tutti e soli gli elementi di *L* che sono multipli di *x* ma non di *y*.

5. Dimostrazioni per induzione

Considera la procedura *g* e dimostra per induzione che il valore dell'espressione *(g L)*, dove *L* è una lista non vuota di numeri naturali positivi, è dato dal minimo comune multiplo (*lcm*) degli elementi di *L*. In particolare:

```
(define g
  (lambda (L) (g-iter (cdr L) (car L)) ))

(define g-iter
  (lambda (L x)
    (if (null? L) x
        (g-iter (cdr L) (lcm (car L) x)))
  ))
```

- Scrivi formalmente la proprietà che intendi dimostrare per induzione.
- Scrivi formalmente la proprietà che esprime il caso base.
- Scrivi formalmente l'ipotesi induttiva.
- Scrivi formalmente la proprietà che si deve dimostrare come passo induttivo.
- Dimostra formalmente il caso base.
- Dimostra formalmente il passo induttivo.

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / A

cognome e nome

1. Procedure in Scheme

(f '(a)) → (a) *(f '(a b)) → (a)*

(f '(a b c)) → (a c) *(f '(a b c d)) → (a c)*

caso generale:

(f '(1 2 3 ... n)) → (1 3 ... k)

dove k = n se n è dispari

k = n–1 altrimenti

2. Realizzazione di strutture dati

Definizione delle procedure:

vedi in fondo

3. Applicazione di strutture dati

Definizione:

```
(define heap?
  (lambda (T)
    (if (singoletto? T)
        #t
        (let ((sin (sottoalbero-sinistro T))
              (des (sottoalbero-destro T))
              )
          (and (heap? sin)
               (heap? des)
               (< (nodo-radice sin) (nodo-radice T))
               (< (nodo-radice des) (nodo-radice T)))
          ))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y lista)
    (cond ((null? lista) null)
          ((or (< (car lista) x) (> (car lista) y))
           (selezione x y (cdr lista)))
          (else (cons (car lista)
                      (selezione x y (cdr lista))))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione: Per ogni lista di naturali L e per ogni numero x $(f\text{-}iter\ L\ x) \rightarrow min(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$ (Dimostrazione per induzione sulla lunghezza di L)	Proprietà dimostrata nel caso base: Per ogni numero x $(f\text{-}iter\ \lambda\ x) \rightarrow min(\{x\}) = x$ (λ è la lista vuota)
Ipotesi induttiva: Per ogni lista di naturali L di lunghezza $\|L\| = n-1$, con $n > 0$, e per ogni numero x $(f\text{-}iter\ L\ x) \rightarrow min(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$	Proprietà dimostrata nel passo induttivo: Per ogni lista di naturali L' di lunghezza $\|L'\| = n$, e per ogni numero y $(f\text{-}iter\ L'\ y) \rightarrow min(\{z \,/\, z \text{ elemento di } L'\} \cup \{x\})$

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / B

cognome e nome

1. Procedure in Scheme

(f '(a)) → () *(f '(a b)) → (b)*

(f '(a b c)) → (b) *(f '(a b c d)) → (b d)*

caso generale:

(f '(1 2 3 ... n)) → (2 4 ... k)

dove k = n se n è pari

k = n–1 altrimenti

2. Realizzazione di strutture dati

Definizione delle procedure:

vedi in fondo

3. Applicazione di strutture dati

Definizione:

```
(define heap?
  (lambda (T)
    (if (singoletto? T)
        #t
        (let ((sin (sottoalbero-sinistro T))
              (des (sottoalbero-destro T))
              )
          (and (heap? sin)
               (heap? des)
               (> (nodo-radice sin) (nodo-radice T))
               (> (nodo-radice des) (nodo-radice T)))
          ))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y lista)
    (cond ((null? lista) null)
          ((and (>= (car lista) x) (<= (car lista) y))
           (selezione x y (cdr lista)))
          (else (cons (car lista)
                      (selezione x y (cdr lista))))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione: Per ogni lista di naturali L e per ogni naturale $x > 0$ $(f\text{-}iter\ L\ x) \rightarrow gcd(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$ (Dimostrazione per induzione sulla lunghezza di L)	Proprietà dimostrata nel caso base: Per ogni naturale $x > 0$ $(f\text{-}iter\ \lambda\ x) \rightarrow gcd(\{x\}) = x$ (λ è la lista vuota)
Ipotesi induttiva: Per ogni lista di naturali L di lunghezza $\|L\| = n-1$, con $n > 0$, e per ogni naturale $x > 0$ $(f\text{-}iter\ L\ x) \rightarrow gcd(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$	Proprietà dimostrata nel passo induttivo: Per ogni lista di naturali L' di lunghezza $\|L'\| = n$, e per ogni naturale $y > 0$ $(f\text{-}iter\ L'\ y) \rightarrow gcd(\{z \,/\, z \text{ elemento di } L'\} \cup \{x\})$

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / C

cognome e nome

1. Procedure in Scheme

(f '(a)) → () *(f '(a b)) → (b)*

(f '(a b c)) → (b) *(f '(a b c d)) → (d b)*

caso generale:

(f '(1 2 3 ... n)) → (k ... 4 2)

dove k = n se n è pari

k = n–1 altrimenti

2. Realizzazione di strutture dati

Definizione delle procedure:

vedi in fondo

3. Applicazione di strutture dati

Definizione:

```
(define trovato?
  (lambda (x T)
    (cond ((= x (nodo-radice T)) #t)
          ((singoletto? T) #f)
          ((< x (nodo-radice T))
           (trovato? x (sottoalbero-sinistro T)))
          (else
           (trovato? x (sottoalbero-destro T)))
          )
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y lista)
    (cond ((null? lista) null)
          ((and (= (remainder (car lista) x) 0)
                (= (remainder (car lista) y) 0))
           (cons (car lista)
                 (selezione x y (cdr lista))))
          (else (selezione x y (cdr lista)))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione:

Per ogni lista di naturali L e per ogni numero x

$(g\text{-}iter\ L\ x) \rightarrow max(\{z\ /\ z\ elemento\ di\ L\} \cup \{x\})$

(Dimostrazione per induzione sulla lunghezza di L)

Proprietà dimostrata nel caso base:

Per ogni numero x

$(g\text{-}iter\ \lambda\ x) \rightarrow max(\{x\}) = x$

(λ è la lista vuota)

Ipotesi induttiva:

Per ogni lista di naturali L di lunghezza $|L| = n-1$, con $n > 0$, e per ogni numero x

$(g\text{-}iter\ L\ x) \rightarrow max(\{z\ /\ z\ elemento\ di\ L\} \cup \{x\})$

Proprietà dimostrata nel passo induttivo:

Per ogni lista di naturali L' di lunghezza $|L'| = n$, e per ogni numero y

$(g\text{-}iter\ L'\ y) \rightarrow max(\{z\ /\ z\ elemento\ di\ L'\} \cup \{x\})$

Corso di Programmazione

II Accertamento del 15 Marzo 2002 / D

cognome e nome

1. Procedure in Scheme

(f '(a)) → (a) *(f '(a b)) → (a)*

(f '(a b c)) → (c a) *(f '(a b c d)) → (c a)*

caso generale:

(f '(1 2 3 ... n)) → (k ... 3 1)

dove k = n se n è dispari

k = n–1 altrimenti

2. Realizzazione di strutture dati

Definizione delle procedure:

vedi in fondo

3. Applicazione di strutture dati

Definizione:

```
(define trovato?
  (lambda (x T)
    (cond ((= x (nodo-radice T)) #t)
          ((singoletto? T) #f)
          ((> x (nodo-radice T))
           (trovato? x (sottoalbero-sinistro T)))
          (else
           (trovato? x (sottoalbero-destro T)))
          )
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define selezione
  (lambda (x y lista)
    (cond ((null? lista) null)
          ((and (= (remainder (car lista) x) 0)
                (not (= (remainder (car lista) y) 0)))
           (cons (car lista)
                 (selezione x y (cdr lista))))
          (else (selezione x y (cdr lista)))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione: Per ogni lista di naturali L e per ogni naturale $x > 0$ $(g\text{-}iter\ L\ x) \rightarrow lcm(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$ (Dimostrazione per induzione sulla lunghezza di L)	Proprietà dimostrata nel caso base: Per ogni naturale $x > 0$ $(g\text{-}iter\ \lambda\ x) \rightarrow lcm(\{x\}) = x$ (λ è la lista vuota)
Ipotesi induttiva: Per ogni lista di naturali L di lunghezza $\|L\| = n-1$, con $n > 0$, e per ogni naturale $x > 0$ $(g\text{-}iter\ L\ x) \rightarrow lcm(\{z \,/\, z \text{ elemento di } L\} \cup \{x\})$	Proprietà dimostrata nel passo induttivo: Per ogni lista di naturali L' di lunghezza $\|L'\| = n$, e per ogni naturale $y > 0$ $(g\text{-}iter\ L'\ y) \rightarrow lcm(\{z \,/\, z \text{ elemento di } L'\} \cup \{x\})$

Esercizio 2

```
(define matrice
  (lambda (r c)
    (if (= r 1)
        (cons (riga c) null)
        (cons (riga c) (matrice (- r 1) c))
        )
    ))

(define riga
  (lambda (c)
    (if (= c 1)
        (cons 0 null)
        (cons 0 (riga (- c 1)))
        )
    ))



(define elemento
  (lambda (M i j)
    (if (= i 1)
        (elemento-riga (car M) j)
        (elemento (cdr M) (- i 1) j)
        )
    ))

(define elemento-riga
  (lambda (R j)
    (if (= j 1)
        (car R)
        (elemento-riga (cdr R) (- j 1))
        )
    ))



(define assegnazione
  (lambda (M i j x)
    (if (= i 1)
        (cons (assegna-riga (car M) j x) (cdr M))
        (cons (car M) (assegnazione (cdr M) (- i 1) j x))
        )
    ))

(define assegna-riga
  (lambda (R j x)
    (if (= j 1)
        (cons x (cdr R))
        (cons (car R) (assegna-riga (cdr R) (- j 1) x))
        )
    ))
```

```
(define numero-righe
  (lambda (M)
    (if (null? M)
        0
        (+ (numero-righe (cdr M)) 1)
        )
    ))

(define numero-colonne
  (lambda (M)
    (lunghezza (car M))
    ))
```