# Corso di Programmazione

**II Accertamento del 27 Marzo 2001 / 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 1 2) (f 2 1) (f 3 3) (f 4 5) (f 5 0)*

e ipotizza il risultato della generica valutazione *(f m n)* .

```
(define f
  (lambda (x y)  ; x, y naturali
    (g (- x 1) y null) ))

(define g
  (lambda (x y z)
    (if (= y 0)  z
        (g x (- y 1) (cons (+ x y) z)) )))
```

**2. Realizzazione di strutture dati**

Descrivi sinteticamente una possibile rappresentazione in Scheme degli insiemi con un numero finito di elementi (qualsiasi). Quindi realizza le seguenti operazioni sugli insiemi.
*insieme-vuoto?* verifica se un dato insieme è vuoto; *rimuovi* assume come valore l'insieme che si ottiene togliendo un elemento dall'insieme argomento, se l'elemento c'è, oppure l'insieme stesso; *unione* realizza l'unione insiemistica.

```
;  Operazioni relative all'ADT "Insiemi"
;  (insieme-vuoto? <insieme>)      :  insieme -> booleano
;  (rimuovi <elemento> <insieme>)  :  elemento x insieme -> insieme
;  (unione <insieme> <insieme>)    :  insieme x insieme -> insieme
;  ecc.
```

**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 due alberi binari sono speculari (uno è l'immagine riflessa dell'altro).

```
;  Operazioni relative all'ADT "Alberi binari"
;  (vuoto? <albero>)                :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> nodo
```

**4. Definizione di procedure in Scheme**

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dato un numero naturale positivo *n*, generare tutti e soli i numeri primi minori o uguali a *n*. Rappresenta la soluzione attraverso una lista ordinata di numeri primi.

**5. Dimostrazioni per induzione**

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)* è dato da:

$$\sum_{x \in L} x^2$$

```
(define f  ; L lista di naturali
  (lambda (L) (g L 0)) )

(define g
  (lambda (L S)
    (if (null? L)  S
        (g (cdr L)
           (+ S (* (car L) (car L))) ) )))
```

In particolare:

- 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 27 Marzo 2001 / 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 1 2)  (f 2 1)  (f 3 3)  (f 4 5)  (f 5 0)*

e ipotizza il risultato della generica valutazione *(f m n)* .

```
(define f
  (lambda (u v)  ; u, v naturali
    (g '() (- (+ u v) 1) v) ))

(define g
  (lambda (t u v)
    (if (= v 0)  t
        (g (cons u t) (- u 1) (- v 1)) )))
```

**2. Realizzazione di strutture dati**

Descrivi sinteticamente una possibile rappresentazione in Scheme degli insiemi con un numero finito di elementi (qualsiasi). Quindi realizza le seguenti operazioni sugli insiemi.
*insieme-vuoto* assume come valore l'insieme vuoto; *rimuovi* calcola l'insieme che si ottiene togliendo un elemento dall'insieme argomento, se l'elemento c'è, oppure l'insieme stesso; *intersezione* realizza l'intersezione insiemistica.

```
;  Operazioni relative all'ADT "Insiemi"
;  (insieme-vuoto)                    :  -> insieme
;  (rimuovi <elemento> <insieme>)     :  elemento x insieme -> insieme
;  (intersezione <insieme> <insieme>) :  insieme x insieme -> insieme
;  ecc.
```

**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 due alberi binari sono identici.

```
;  Operazioni relative all'ADT "Alberi binari"
;  (vuoto? <albero>)                :  albero -> booleano
;  (sottoalbero-sinistro <albero>)  :  albero -> albero
;  (sottoalbero-destro <albero>)    :  albero -> albero
;  (nodo-radice <albero>)           :  albero -> nodo
```

**4. Definizione di procedure in Scheme**

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dato un numero naturale positivo *n*, generare tutti e soli i numeri minori o uguali a *n* che *non* sono numeri primi. Rappresenta la soluzione attraverso una lista ordinata.

**5. Dimostrazioni per induzione**

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)* è dato da:

$$\sum_{x \in L} x$$

```
(define f
  (lambda (L)  ; L lista di naturali
    (g 0 L) ))

(define g
  (lambda (A L)
    (if (null? L)  A
        (g (+ A (car L)) (cdr L)) )))
```

In particolare:

- 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 27 Marzo 2001 / 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 1 2) (f 2 1) (f 3 3) (f 4 5) (f 5 0)*

e ipotizza il risultato della generica valutazione *(f m n)* .

```
(define f
  (lambda (x y)  ; x, y naturali
    (g x y null) ))

(define g
  (lambda (x y z)
    (if (= y 0)  z
        (g (+ x 1) (- y 1) (cons x z)) )))
```

**2. Realizzazione di strutture dati**

Descrivi sinteticamente una possibile rappresentazione in Scheme degli insiemi con un numero finito di elementi (qualsiasi). Quindi realizza le seguenti operazioni sugli insiemi.
*insieme-vuoto?* verifica se un dato insieme è vuoto; *aggiungi* assume come valore l'insieme che si ottiene aggiungendo un elemento all'insieme argomento, se non c'è già, oppure l'insieme stesso; *differrenza* assume come valore l'insieme degli elementi che appartengono al primo argomento (che è un insieme) ma non al secondo.

```
;  Operazioni relative all'ADT "Insiemi"
;  (insieme-vuoto? <insieme>)        :  insieme -> booleano
;  (aggiungi <elemento> <insieme>)   :  elemento x insieme -> insieme
;  (differenza <insieme> <insieme>)  :  insieme x insieme -> insieme
;  ecc.
```

**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 calcolare l'altezza di un albero binario, cioè il numero di nodi del più lungo cammino dalla radice a una foglia.

```
;  Operazioni relative all'ADT "Alberi binari"
;  (vuoto? <albero>)                 :  albero -> booleano
;  (sottoalbero-sinistro <albero>)   :  albero -> albero
;  (sottoalbero-destro <albero>)     :  albero -> albero
;  (nodo-radice <albero>)            :  albero -> nodo
```

**4. Definizione di procedure in Scheme**

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dato un numero naturale positivo *n*, calcolare la fattorizzazione di *n* in fattori primi. Rappresenta la fattorizzazione attraverso una lista ordinata di numeri primi in cui ciascun fattore primo compare esattamente tante volte quanto è il valore del relativo esponente nella fattorizzazione di *n.*

**5. Dimostrazioni per induzione**

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)* è dato da:

$$\prod_{x \in L} (x+1)$$

```
(define f  ; L lista di naturali
  (lambda (L) (g 1 L)) )

(define g
  (lambda (Q L)
    (if (null? L)  Q
        (g (* (+ (car L) 1) Q)
           (cdr L)) )))
```

In particolare:

- 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 27 Marzo 2001 / 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 1 2) (f 2 1) (f 3 3) (f 4 5) (f 5 0)*

e ipotizza il risultato della generica valutazione *(f m n)* .

```
(define f
  (lambda (u v)  ; u, v naturali
    (g '() u (- (+ u v) 1)) ))

(define g
  (lambda (t u v)
    (if (> u v)  t
        (g (cons u t) (+ u 1) v) )))
```

**2. Realizzazione di strutture dati**

Descrivi sinteticamente una possibile rappresentazione in Scheme degli insiemi con un numero finito di elementi (qualsiasi). Quindi realizza le seguenti operazioni sugli insiemi.
*insieme-vuoto* assume come valore l'insieme vuoto; *aggiungi* valuta l'insieme che si ottiene aggiungendo un elemento all'insieme argomento, se non c'è già, oppure l'insieme stesso; *diff-simmetrica* (differenza simmetrica) assume come valore l'insieme degli elementi che appartengono a uno dei due argomenti (che sono insiemi) ma non ad entrambi.

```
;  Operazioni relative all'ADT "Insiemi"
;  (insieme-vuoto)                          :  -> insieme
;  (aggiungi <elemento> <insieme>)          :  elemento x insieme -> insieme
;  (diff-simmetrica <insieme> <insieme>)    :  insieme x insieme -> insieme
;  ecc.
```

**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 calcolare il numero complessivo di nodi di un albero binario.

```
;  Operazioni relative all'ADT "Alberi binari"
;  (vuoto? <albero>)                 :  albero -> booleano
;  (sottoalbero-sinistro <albero>)   :  albero -> albero
;  (sottoalbero-destro <albero>)     :  albero -> albero
;  (nodo-radice <albero>)            :  albero -> nodo
```

**4. Definizione di procedure in Scheme**

Definisci formalmente una procedura in Scheme per risolvere il seguente problema: dato un numero naturale positivo *n*, calcolare la fattorizzazione di *n* in fattori primi. Rappresenta la fattorizzazione attraverso una lista di coppie, dove ciascun fattore primo risulta associato al relativo esponente nella fattorizzazione di *n*.

**5. Dimostrazioni per induzione**

Considera la procedura *f* e dimostra per induzione che il valore dell'espressione *(f L)* è dato da:

$$\prod_{x \in L} x$$

```
(define f
  (lambda (L)  ; x lista di naturali
    (g L 1) ))

(define g
  (lambda (L P)
    (if (null? L)  P
        (g (cdr L) (* (car L) P)) )))
```

In particolare:

- 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 27 Marzo 2001 / A**

cognome e nome

**1. Procedure in Scheme**

*(f 1 2) → (1 2)  (f 2 1) → (2)  (f 3 3) → (3 4 5)*

*(f 4 5) → (4 5 6 7 8)  (f 5 0) → ()*

caso generale:

*(f m n) → (m m+1 . . . m+n–1)*

**2. Realizzazione di strutture dati**

Definizione delle procedure:

Vedi la realizzazione completa di una struttura dati "Insiemi" presentata dopo queste schede. Gli insiemi sono rappresentati da liste senza ripetizioni.

**3. Applicazione di strutture dati**

Definizione:

```
(define speculari?
  (lambda (albero1 albero2)
    (cond ((and (vuoto? albero1) (vuoto? albero2)) #t)
          ((or  (vuoto? albero1) (vuoto? albero2)) #f)
          (else
           (and
            (equal? (nodo-radice albero1)
                    (nodo-radice albero2))
            (speculari? (sottoalbero-sinistro albero1)
                        (sottoalbero-destro   albero2))
            (speculari? (sottoalbero-destro   albero1)
                        (sottoalbero-sinistro albero2))
           )))
    ))
```

## 4. Definizione di procedure in Scheme

Definizione:

```
(define lista-primi
  (lambda (n)
    (numeri-primi 2 n)
    ))

(define numeri-primi
  (lambda (k n)
    (cond ((> k n) null)
          ((primo-da? 2 k) (cons k (numeri-primi (+ k 1) n)))
          (else (numeri-primi (+ k 1) n))
    )))

(define primo-da?
  (lambda (d n)
    (cond ((= d n) #t)
          ((> (remainder n d) 0) (primo-da? (+ d 1) n))
          (else #f)
    )))
```

## 5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione:

Per ogni lista di naturali $L$ e per ogni naturale $S$

$$(g\ L\ S) \rightarrow S + \sum_{x \in L} x^2$$

(Dimostrazione per induzione sulla lunghezza di $L$)

Proprietà dimostrata nel caso base:

Per ogni naturale $S$

$$(g\ '()\ S) \rightarrow S$$

Ipotesi induttiva:

Per ogni lista di naturali $L$ di lunghezza $|L| < n$, con $n > 0$, e per ogni naturale $S$

$$(g\ L\ S) \rightarrow S + \sum_{x \in L} x^2$$

Proprietà dimostrata nel passo induttivo:

Per ogni lista di naturali $L$ di lunghezza $|L| = n$, e per ogni naturale $S$

$$(g\ L\ S) \rightarrow S + \sum_{x \in L} x^2$$

# Corso di Programmazione

**II Accertamento del 27 Marzo 2001 / B**

cognome e nome

**1. Procedure in Scheme**

*(f 1 2) → (1 2)  (f 2 1) → (2)  (f 3 3) → (3 4 5)*

*(f 4 5) → (4 5 6 7 8)  (f 5 0) → ()*

caso generale:

*(f m n) → (m m+1 . . . m+n–1)*

**2. Realizzazione di strutture dati**

Definizione delle procedure:

Vedi la realizzazione completa di una struttura dati "Insiemi" presentata dopo queste schede. Gli insiemi sono rappresentati da liste senza ripetizioni.

**3. Applicazione di strutture dati**

Definizione:

```
(define identici?
  (lambda (albero1 albero2)
    (cond ((and (vuoto? albero1) (vuoto? albero2)) #t)
          ((or  (vuoto? albero1) (vuoto? albero2)) #f)
          (else
           (and
            (equal? (nodo-radice albero1)
                    (nodo-radice albero2))
            (identici? (sottoalbero-sinistro albero1)
                       (sottoalbero-sinistro albero2))
            (identici? (sottoalbero-destro albero1)
                       (sottoalbero-destro albero2))
          )))
    ))
```

## 4. Definizione di procedure in Scheme

Definizione:

```
(define lista-non-primi
  (lambda (n) (numeri-non-primi 2 n)) )

(define numeri-non-primi
  (lambda (k n)
    (cond ((> k n) null)
          ((primo-da? 2 k) (numeri-non-primi (+ k 1) n))
          (else (cons k (numeri-non-primi (+ k 1) n)))
    )))

(define primo-da?
  (lambda (d n)
    (cond ((= d n) #t)
          ((> (remainder n d) 0) (primo-da? (+ d 1) n))
          (else #f)
    )))
```

## 5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione:

Per ogni lista di naturali $L$ e per ogni naturale $A$

$$(g\ A\ L) \rightarrow A + \sum_{x \in L} x$$

(Dimostrazione per induzione sulla lunghezza di $L$)

Proprietà dimostrata nel caso base:

Per ogni naturale $A$

$$(g\ A\ '()) \rightarrow A$$

Ipotesi induttiva:

Per ogni lista di naturali $L$ di lunghezza $|L| < n$, con $n > 0$, e per ogni naturale $A$

$$(g\ A\ L) \rightarrow A + \sum_{x \in L} x$$

Proprietà dimostrata nel passo induttivo:

Per ogni lista di naturali $L$ di lunghezza $|L| = n$, e per ogni naturale $A$

$$(g\ A\ L) \rightarrow A + \sum_{x \in L} x$$

# Corso di Programmazione

**II Accertamento del 27 Marzo 2001 / C**

cognome e nome

**1. Procedure in Scheme**

*(f 1 2) → (2 1)  (f 2 1) → (2)  (f 3 3) → (5 4 3)*

*(f 4 5) → (8 7 6 5 4)  (f 5 0) → ()*

caso generale:

*(f m n) → (m+n–1  m+n–2  . . .  m)*

**2. Realizzazione di strutture dati**

Definizione delle procedure:

Vedi la realizzazione completa di una struttura dati "Insiemi" presentata dopo queste schede. Gli insiemi sono rappresentati da liste senza ripetizioni.

**3. Applicazione di strutture dati**

Definizione:

```
(define max
  (lambda (x y)
    (if (< x y) y x)
    ))

(define altezza
  (lambda (albero)
    (if (vuoto? albero)
        0
        (+ 1
           (max (altezza (sottoalbero-sinistro albero))
                (altezza (sottoalbero-destro   albero)))
        ))
    ))
```

## 4. Definizione di procedure in Scheme

Definizione:

```
(define fattorizzazione
  (lambda (n)
    (fattorizzazione-da 2 n)
    ))

(define fattorizzazione-da
  (lambda (k n)
    (cond ((= n 1) null)
          ((= (remainder n k) 0)
           (cons k (fattorizzazione-da k (quotient n k))))
          (else (fattorizzazione-da (+ k 1) n))
    )))
```

## 5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione:

Per ogni lista di naturali $L$ e per ogni naturale $Q$

$$(g\ Q\ L) \rightarrow Q \times \prod_{x \in L} (x+1)$$

(Dimostrazione per induzione sulla lunghezza di $L$)

Proprietà dimostrata nel caso base:

Per ogni naturale $Q$

$$(g\ Q\ '()) \rightarrow Q$$

Ipotesi induttiva:

Per ogni lista di naturali $L$ di lunghezza $|L| < n$, con $n > 0$, e per ogni naturale $Q$

$$(g\ Q\ L) \rightarrow Q \times \prod_{x \in L} (x+1)$$

Proprietà dimostrata nel passo induttivo:

Per ogni lista di naturali $L$ di lunghezza $|L| = n$, e per ogni naturale $Q$

$$(g\ Q\ L) \rightarrow Q \times \prod_{x \in L} (x+1)$$

# Corso di Programmazione

**II Accertamento del 27 Marzo 2001 / D**

cognome e nome

**1. Procedure in Scheme**

*(f 1 2) → (2 1)  (f 2 1) → (2)  (f 3 3) → (5 4 3)*

*(f 4 5) → (8 7 6 5 4)  (f 5 0) → ()*

caso generale:

*(f m n) → (m+n–1 m+n–2 . . . m)*

**2. Realizzazione di strutture dati**

Definizione delle procedure:

Vedi la realizzazione completa di una struttura dati "Insiemi" presentata dopo queste schede. Gli insiemi sono rappresentati da liste senza ripetizioni.

**3. Applicazione di strutture dati**

Definizione:

```
(define numero-nodi
  (lambda (albero)
    (if (vuoto? albero)
        0
        (+ 1
           (numero-nodi (sottoalbero-sinistro albero))
           (numero-nodi (sottoalbero-destro   albero))
        ))
    ))
```

## 4. Definizione di procedure in Scheme

Definizione:

```
(define fattorizzazione
  (lambda (n) (fattorizzazione-da 2 n)) )

(define fattorizzazione-da
  (lambda (k n)
    (cond ((= n 1) null)
          ((= (remainder n k) 0)
           (cons (cons k (esponente k n))
                 (fattorizzazione-da (+ k 1) (riduci n k))))
          (else (fattorizzazione-da (+ k 1) n))
    )))

(define esponente
  (lambda (p n)
    (if (= (remainder n p) 0)
        (+ (esponente p (quotient n p)) 1)
        0
    )))

(define riduci
  (lambda (n p)
    (if (= (remainder n p) 0)
        (riduci (quotient n p) p)
        n
    )))
```

## 5. Dimostrazioni per induzione

Dimostrazioni su foglio allegato.

Proprietà da dimostrare per induzione:

Per ogni lista di naturali $L$ e per ogni naturale $P$

$$(g\ L\ P) \rightarrow P \times \prod_{x \in L} x$$

(Dimostrazione per induzione sulla lunghezza di $L$)

Proprietà dimostrata nel caso base:

Per ogni naturale $P$

$$(g\ '()\ P) \rightarrow P$$

Ipotesi induttiva:

Per ogni lista di naturali $L$ di lunghezza $|L| < n$, con $n > 0$, e per ogni naturale $P$

$$(g\ L\ P) \rightarrow P \times \prod_{x \in L} x$$

Proprietà dimostrata nel passo induttivo:

Per ogni lista di naturali $L$ di lunghezza $|L| = n$, e per ogni naturale $P$

$$(g\ L\ P) \rightarrow P \times \prod_{x \in L} x$$

```
;  Realizzazione dell'ADT "Insiemi"
;
;  (insieme-vuoto)                          :  -> insieme
;  (insieme-vuoto? <insieme>)               :  insieme -> booleano
;  (aggiungi <elemento> <insieme>)          :  elemento x insieme -> insieme
;  (rimuovi <elemento> <insieme>)           :  elemento x insieme -> insieme
;  (unione <insieme> <insieme>)             :  insieme x insieme -> insieme
;  (intersezione <insieme> <insieme>)       :  insieme x insieme -> insieme
;  (differenza <insieme> <insieme>)         :  insieme x insieme -> insieme
;  (diff-simmetrica <insieme> <insieme>)    :  insieme x insieme -> insieme

(define insieme-vuoto (lambda () null))

(define insieme-vuoto? null?)

(define appartiene?
  (lambda (el ins)
    (cond ((null? ins) #f)
          ((equal? (car ins) el) #t)
          (else (appartiene? el (cdr ins)))
    )))

(define aggiungi
  (lambda (el ins)
    (cond ((null? ins) (cons el null))
          ((equal? (car ins) el) ins)
          (else (cons (car ins) (aggiungi el (cdr ins))))
    )))

(define rimuovi
  (lambda (el ins)
    (cond ((null? ins) null)
          ((equal? (car ins) el) (cdr ins))
          (else (cons (car ins) (rimuovi el (cdr ins))))
    )))

(define unione
  (lambda (ins1 ins2)
    (if (null? ins1)
        ins2
        (unione (cdr ins1) (aggiungi (car ins1) ins2))
    )))

(define intersezione
  (lambda (ins1 ins2)
    (cond ((null? ins1) null)
          ((appartiene? (car ins1) ins2)
           (cons (car ins1) (intersezione (cdr ins1) ins2)))
          (else (intersezione (cdr ins1) ins2))
    )))

(define differenza
  (lambda (ins1 ins2)
    (if (null? ins2)
        ins1
        (differenza (rimuovi (car ins2) ins1) (cdr ins2))
    )))

(define diff-simmetrica
  (lambda (ins1 ins2)
    (cond ((null? ins1) ins2)
          ((appartiene? (car ins1) ins2)
           (diff-simmetrica (cdr ins1) (rimuovi (car ins1) ins2)))
          (else (cons (car ins1) (diff-simmetrica (cdr ins1) ins2)))
    )))
```