Corso di Programmazione

I Accertamento del 14 Dicembre 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. Dovrai poi consegnare questo testo e il foglio con le soluzioni, avendo cura di scrivere il tuo nome su ciascun foglio.

1. Procedure in Scheme

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

(f 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (n)  ; n naturale
    (f-iter n 1) ))

(define f-iter
  (lambda (k p)
    (if (= k 0)
        p
        (f-iter (- k 1) (* p k))
      ) ))
```

2. Procedure in Scheme

Completa il programma in Scheme a fianco per verificare se due funzioni *f*, *g* dai naturali ai naturali sono uguali nell'intervallo *[a, b]*.

```
(define uguali
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((= _____ (g a))
           (uguali f g _____ _____ ))
          (else _____ )
      ) ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per calcolare quante volte occorre la cifra zero nella rappresentazione decimale di un dato numero naturale *n*.

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Data una funzione *g* definita sull'insieme dei naturali e a valori naturali, e dati due numeri naturali *a*, *b*, si vuole conoscere il massimo incremento del valore della funzione fra due punti consecutivi dell'intervallo *[a, b]*.

5. Dimostrazioni per induzione

Considera la procedura *f* e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(f n)* è dato da:

$$2\,(n2^n - 2^n + 1)$$

```
(define f
  (lambda (n)    ; n naturale
    (if (= n 0)
        0
        (+ (f (- n 1))
           (* n (expt 2 n)))
      ) ))
```

In particolare:
- 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.

6. Ricorsione di coda e invarianti

Esprimi formalmente, in funzione dei parametri, la quantità invariante relativa alla procedura ricorsiva di coda dell'esercizio 1.

Corso di Programmazione

I Accertamento del 14 Dicembre 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. Dovrai poi consegnare questo testo e il foglio con le soluzioni, avendo cura di scrivere il tuo nome su ciascun foglio.

1. Procedure in Scheme

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

(f 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (n)  ; n naturale
    (f-iter 1 n 1) ))

(define f-iter
  (lambda (i k p)
    (if (> i k)
        p
        (f-iter (+ i 1) k (* p i))
    ) ))
```

2. Procedure in Scheme

Completa il programma in Scheme a fianco per verificare se due funzioni *f*, *g* dai naturali ai naturali sono diverse nell'intervallo *[a, b]*.

```
(define diverse
  (lambda (f g a b)  ; a <= b
    (if (= _____ (g a))
        (if (< a b) (diverse f g _____ _____ )
                    #f)
        _____
    ) ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per calcolare quante volte occorre la cifra uno nella rappresentazione binaria di un dato numero naturale *n*.

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Data una funzione *f* definita sull'insieme dei naturali e a valori naturali, e dati due numeri naturali *a*, *b*, si vuole conoscere il massimo decremento (in valore assoluto) del valore della funzione fra due punti consecutivi dell'intervallo *[a, b]*.

5. Dimostrazioni per induzione

Considera la procedura *g* e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(g n)* è dato da:

$$2\,(n2^n - 2^n + 1)$$

```
(define g
  (lambda (n)    ; n naturale
    (if (= n 0)
        0
        (* 2 (+ (g (- n 1))
                (- (expt 2 n) 1)))
    ) ))
```

In particolare:
- 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.

6. Ricorsione di coda e invarianti

Esprimi formalmente, in funzione dei parametri, la quantità invariante relativa alla procedura ricorsiva di coda dell'esercizio 1.

Corso di Programmazione

I Accertamento del 14 Dicembre 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. Dovrai poi consegnare questo testo e il foglio con le soluzioni, avendo cura di scrivere il tuo nome su ciascun foglio.

1. Procedure in Scheme

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

(f 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (n)
    (f-iter 0 n) ))

(define f-iter
  (lambda (s k)
    (if (= k 0)
        s
        (f-iter (+ s k) (- k 1))
    ) ))
```

2. Procedure in Scheme

Completa il programma in Scheme a fianco per verificare se, date le funzioni *f*, *g* dai naturali ai naturali, *f* è sempre maggiore di *g* nell'intervallo *[a, b]*.

```
(define maggiore
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((<= _____ (g a)) _____ )
          (else (maggiore f g _____ _____ ))
    ) ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per calcolare la posizione, partendo dalla cifra meno significativa e procedendo verso sinistra, della prima cifra uno nella rappresentazione binaria di un dato numero naturale *n*.

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Data una funzione *g* definita sull'insieme dei naturali e a valori naturali, e dati due naturali *a*, *b*, si vuole verificare se la funzione è crescente nell'intervallo *[a, b]*.

5. Dimostrazioni per induzione

Considera la procedura *f* e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(f n)* è dato da:

$$\frac{(2n-1)3^{n+1} + 3}{4}$$

```
(define f
  (lambda (n)    ; n naturale
    (if (= n 0)
        0
        (+ (f (- n 1))
           (* n (expt 3 n)))
    ) ))
```

In particolare:
- 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.

6. Ricorsione di coda e invarianti

Esprimi formalmente, in funzione dei parametri, la quantità invariante relativa alla procedura ricorsiva di coda dell'esercizio 1.

Corso di Programmazione

I Accertamento del 14 Dicembre 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. Dovrai poi consegnare questo testo e il foglio con le soluzioni, avendo cura di scrivere il tuo nome su ciascun foglio.

1. Procedure in Scheme

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

(f 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (n)
    (f-iter 1 n 0) ))

(define f-iter
  (lambda (x y s)
    (if (> x y)
        s
        (f-iter (+ x 1) y (+ s x))
    ) ))
```

2. Procedure in Scheme

Completa il programma in Scheme a fianco per verificare se, date le funzioni *f*, *g* dai naturali ai naturali, *f* è sempre minore di *g* nell'intervallo *[a, b]*.

```
(define minore
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((< (f a) _____ )
           (minore f _____ _____ b))
          (else _____ )
    ) ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per calcolare la posizione, partendo dalla cifra meno significativa e procedendo verso sinistra, della prima cifra diversa da zero nella rappresentazione decimale di un dato numero naturale *n*.

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Data una funzione *f* definita sull'insieme dei naturali e a valori naturali, e dati due naturali *a*, *b*, si vuole verificare se la funzione è decrescente nell'intervallo *[a, b]*.

5. Dimostrazioni per induzione

Considera la procedura *g* e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(g n)* è dato da:

$$\frac{(2n-1)3^{n+1} + 3}{4}$$

```
(define g
 (lambda (n)    ; n naturale
  (if (= n 0)
      0
      (* 3 (+ (g (- n 1))
              (/ (- (expt 3 n) 1) 2)))
  ) ))
```

In particolare:
- 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.

6. Ricorsione di coda e invarianti

Esprimi formalmente, in funzione dei parametri, la quantità invariante relativa alla procedura ricorsiva di coda dell'esercizio 1.

I Accertamento di Programmazione / A

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

(f 0) → *1* *(f 1)* → *1* *(f 2)* → *2*

(f 3) → *6* *(f 4)* → *24* *(f 5)* → *120*

caso generale:

(f n) → *n!*

2. Procedure in Scheme

Definizione completa:

```
(define uguali
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((= (f a) (g a))
           (uguali f g (+ a 1) b))
          (else #f)
          )
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define quante-cifre-zero
  (lambda (n)  ; n naturale
    (cond ((= n 0) 1)
          ((< n 10) 0)
          (else
           (let ((contributo-ultima-cifra
                  (if (= (remainder n 10) 0) 1 0)))
             (+ (quante-cifre-zero (quotient n 10))
                contributo-ultima-cifra)
          )))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define massimo-incremento
  (lambda (g a b)  ; a < b
    (let ((incr (- (g (+ a 1)) (g a))))
      (if (= (+ a 1) b)
          incr
          (max (massimo-incremento g (+ a 1) b) incr)
      ))
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni del caso base e del passo induttivo su foglio allegato.

Proprietà dimostrata nel caso base:

$$(f\ 0) \rightarrow 2\,(0 \cdot 2^0 - 2^0 + 1)$$

Ipotesi induttiva: *per* $n > 0$

$$(f\ n-1) \rightarrow 2\,((n-1)2^{n-1} - 2^{n-1} + 1)$$

Proprietà dimostrata nel passo induttivo: *per* $n > 0$

$$(f\ n) \rightarrow 2\,(n2^n - 2^n + 1)$$

Dimostrazione:

$$(f\ n) \rightarrow (+\ (f\ n-1)\ (*\ n\ (expt\ 2\ n)))$$

$$\rightarrow (+\ \ 2 \cdot ((n-1)2^{n-1} - 2^{n-1} + 1)\ \ n \cdot 2^n)$$

$$\rightarrow (n-1)2^n - 2^n + 2\ \ +\ \ n \cdot 2^n$$

$$= 2\,(n2^n - 2^n + 1)$$

6. Ricorsione di coda e invarianti

Quantità invariante: (relativo alla procedura *f-iter*)

$$p \cdot k!$$

I Accertamento di Programmazione / B

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

(f 0) → *1* *(f 1)* → *1* *(f 2)* → *2*

(f 3) → *6* *(f 4)* → *24* *(f 5)* → *120*

caso generale:

(f n) → *n!*

2. Procedure in Scheme

Definizione completa:

```
(define diverse
  (lambda (f g a b)  ; a <= b
    (if (= (f a) (g a))
        (if (< a b) (diverse f g (+ a 1) b)
                    #f)
        #t
        )
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define quanti-bit-uno
  (lambda (n)  ; n naturale
    (cond ((= n 0) 0)
          ((= n 1) 1)
          (else
           (let ((contributo-ultimo-bit
                   (if (= (remainder n 2) 1) 1 0)))
             (+ (quanti-bit-uno (quotient n 2))
                contributo-ultimo-bit)
           )))
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define massimo-decremento
  (lambda (f a b)  ; a < b, incr. = decr. negativo
    (let ((decr (- (f a) (f (+ a 1)))))
      (if (= (+ a 1) b)
          decr
          (max (massimo-decremento f (+ a 1) b) decr)
      ))
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni del caso base e del passo induttivo su foglio allegato.

Proprietà dimostrata nel caso base:

$$(g\ 0) \rightarrow 2\,(0 \cdot 2^0 - 2^0 + 1)$$

Ipotesi induttiva: *per* $n > 0$

$$(g\ n-1) \rightarrow 2\,((n-1)2^{n-1} - 2^{n-1} + 1)$$

Proprietà dimostrata nel passo induttivo: *per* $n > 0$

$$(g\ n) \rightarrow 2\,(n2^n - 2^n + 1)$$

Dimostrazione:

$$(g\ n) \rightarrow (*\ 2\ (+\ (g\ n-1)\ (-\ (expt\ 2\ n)\ 1)))$$

$$\rightarrow (*\ 2\ (+\ 2 \cdot ((n-1)2^{n-1} - 2^{n-1} + 1)\ \ 2^n - 1))$$

$$\rightarrow 2 \cdot ((n-1)2^n - 2^n + 2\ +\ 2^n - 1)$$

$$= 2\,(n2^n - 2^n + 1)$$

6. Ricorsione di coda e invarianti

Quantità invariante: (relativo alla procedura *f-iter*)

$$p \cdot \frac{k!}{(i-1)!}$$

I Accertamento di Programmazione / C

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

$(f\ 0) \rightarrow 0$ $(f\ 1) \rightarrow 1$ $(f\ 2) \rightarrow 3$

$(f\ 3) \rightarrow 6$ $(f\ 4) \rightarrow 10$ $(f\ 5) \rightarrow 15$

caso generale:

$(f\ n) \rightarrow \frac{n\ (n+1)}{2}$

2. Procedure in Scheme

Definizione completa:

```
(define maggiore
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((<= (f a) (g a)) #f)
          (else (maggiore f g (+ a 1) b))
          )
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define primo-bit-uno
  (lambda (n)  ; n > 0 naturale
    (if (= (remainder n 2) 1)
        0
        (+ (primo-bit-uno (quotient n 2)) 1)
        )
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define crescente
  (lambda (g a b)  ; a < b
    (cond ((>= (g a) (g (+ a 1))) #f)
          ((= (+ a 1) b) #t)
          (else (crescente g (+ a 1) b))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni del caso base e del passo induttivo su foglio allegato.

Proprietà dimostrata nel caso base:

$$(f\ 0) \rightarrow \frac{(2\cdot 0 - 1)3^1 + 3}{4}$$

Ipotesi induttiva: *per* $n > 0$

$$(f\ n{-}1) \rightarrow \frac{(2n-3)3^n + 3}{4}$$

Proprietà dimostrata nel passo induttivo: *per* $n > 0$

$$(f\ n) \rightarrow \frac{(2n-1)3^{n+1} + 3}{4}$$

Dimostrazione:

$$(f\ n) \rightarrow (+\ (f\ n{-}1)\ (*\ n\ (expt\ 3\ n)))$$

$$\rightarrow (+\ \frac{(2n-3)3^n + 3}{4}\ \ n\cdot 3^n)$$

$$\rightarrow \frac{(2n + 4n - 3)3^n + 3}{4}$$

$$= \frac{(2n-1)3^{n+1} + 3}{4}$$

6. Ricorsione di coda e invarianti

Quantità invariante: (relativo alla procedura *f-iter*)

$$s + \frac{k \cdot (k+1)}{2}$$

I Accertamento di Programmazione / D

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

$(f\ 0) \rightarrow 0$ $(f\ 1) \rightarrow 1$ $(f\ 2) \rightarrow 3$

$(f\ 3) \rightarrow 6$ $(f\ 4) \rightarrow 10$ $(f\ 5) \rightarrow 15$

caso generale:

$(f\ n) \rightarrow \frac{n\ (n+1)}{2}$

2. Procedure in Scheme

Definizione completa:

```
(define minore
  (lambda (f g a b)
    (cond ((> a b) #t)
          ((< (f a) (g a))
           (minore f g (+ a 1) b))
          (else #f)
          )
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define prima-cifra-non-zero
  (lambda (n)  ; n > 0 naturale
    (if (> (remainder n 10) 0)
        0
        (+ (prima-cifra-non-zero (quotient n 10)) 1)
        )
    ))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define decrescente
  (lambda (f a b)  ; a < b
    (cond ((<= (f a) (f (+ a 1))) #f)
          ((= (+ a 1) b) #t)
          (else (decrescente f (+ a 1) b))
          )
    ))
```

5. Dimostrazioni per induzione

Dimostrazioni del caso base e del passo induttivo su foglio allegato.

Proprietà dimostrata nel caso base:

$$(g\ 0) \rightarrow \frac{(2\cdot 0 - 1)3^1 + 3}{4}$$

Ipotesi induttiva: *per* $n > 0$

$$(g\ n{-}1) \rightarrow \frac{(2n-3)3^n + 3}{4}$$

Proprietà dimostrata nel passo induttivo: *per* $n > 0$

$$(g\ n) \rightarrow \frac{(2n-1)3^{n+1} + 3}{4}$$

Dimostrazione:

$$(g\ n) \rightarrow (*\ 3\ (+\ (g\ n{-}1)\ (/\ (-\ (expt\ 3\ n)\ 1)\ 2)))$$

$$\rightarrow (*\ 3\ (+\ \frac{(2n-3)3^n + 3}{4}\ \ \frac{3^n-1}{2}))$$

$$\rightarrow 3 \cdot \frac{(2n-3)3^n+3 + 2\cdot 3^n-2}{4}$$

$$= \frac{(2n-1)3^{n+1} + 3}{4}$$

6. Ricorsione di coda e invarianti

Quantità invariante: (relativo alla procedura *f-iter*)

$$s + \frac{y\cdot(y+1)}{2} - \frac{x\cdot(x-1)}{2}$$