Corso di Programmazione

I Accertamento del 21 Dicembre 2000 / 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 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (x)     ; x naturale
    (if (< x 2)
        x
        (+ (f (- x 2)) (* 4 (- x 1)))
        )
    ))
```

2. Procedure in Scheme
Completa il programma in Scheme a fianco per calcolare il minimo comune multiplo (*mcm*) di due numeri naturali positivi.

```
(define mcm (lambda (x y) (min-multiplo x x y)))

(define min-multiplo
  (lambda (m x y)
    (if (= __________ 0)
        m
        ( __________   (+ m x)   _____   _____ ) )))
```

3. Definizione di procedure in Scheme
Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Dato un numero naturale *n*, calcolare il numero di divisori distinti (diversi da *1* e *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 si annulla 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:

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

```
(define f
  (lambda (n)     ; n > 0 naturale
    (if (= n 1)
        0
        (+ (f (- n 1))
           (* n (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
Definisci formalmente un programma in Scheme basato sulla ricorsione di coda (approccio iterativo) equivalente alla procedura g.

```
(define g
  (lambda (n)     ; n naturale
    (if (< n 2)
        1
        (* (g (- n 2)) (* n n))
        )
    ))
```

Corso di Programmazione

I Accertamento del 21 Dicembre 2000 / 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 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (x)     ; x naturale
    (if (< x 2)
        x
        (+ (f (- x 2))
           (+ (* (* 6 x) (- x 2)) 8))
        )
    ))
```

2. Procedure in Scheme
Completa il programma in Scheme a fianco per calcolare il minimo comune multiplo (*mcm*) di due numeri naturali positivi.

```
(define mcm (lambda (x y) (min-multiplo x y 1)))

(define min-multiplo
  (lambda (x y k)
    (if (= __________ 0)
        (* k y)
        ( ___________  _____  _____  (+ k 1)) )))
```

3. Definizione di procedure in Scheme
Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Dato un numero naturale *n*, calcolare il più grande divisore di *n*, diverso da *1* e *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 nell'intervallo *[a, b]* esistono soluzioni dell'equazione $f(x) = x$.

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:

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

```
(define f
  (lambda (n)     ; n > 0 naturale
    (if (= n 1)
        0
        (+ (f (- n 1))
           (* (- 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
Definisci formalmente un programma in Scheme basato sulla ricorsione di coda (approccio iterativo) equivalente alla procedura *g*.

```
(define g
  (lambda (n)     ; n naturale
    (if (< n 2)
        1
        (* (g (- n 2)) (* 2 n))
        )
    ))
```

Corso di Programmazione

I Accertamento del 21 Dicembre 2000 / 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 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (x)     ; x naturale
    (if (< x 2)
        (+ x 1)
        (* (f (- x 2)) 4)
        )
    ))
```

2. Procedure in Scheme
Completa il programma in Scheme a fianco per calcolare la parte intera della radice k-ima (*rad*) di un numero naturale n.

```
(define rad (lambda (k n) (max-rad k n 1)))

(define max-rad
  (lambda (k n r)
    (if (> __________ n)
        (- r 1)
        ( __________  _____  _____  (+ r 1)) )))
```

3. Definizione di procedure in Scheme
Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Dato un numero naturale n, verificare se n è un fattoriale ($n = k!$).

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 punto dell'intervallo $[a, b]$ in cui la funzione assume il valore massimo.

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:

$$\frac{(2n + 1)(n + 1)n}{6}$$

```
(define f
  (lambda (n)     ; n > 0 naturale
    (if (= n 1)
        1
        (+ (f (- n 1)) (* n 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
Definisci formalmente un programma in Scheme basato sulla ricorsione di coda (approccio iterativo) equivalente alla procedura g.

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

Corso di Programmazione

I Accertamento del 21 Dicembre 2000 / 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 0) (f 1) (f 2) (f 3) (f 4) (f 5)

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

```
(define f
  (lambda (x)     ; x naturale
    (if (< x 2)
        1
        (* (f (- x 2)) (- (* x x) x))
        )
    ))
```

2. Procedure in Scheme

Completa il programma in Scheme a fianco per calcolare la parte intera della radice *k*-ima (*rad*) di un numero naturale *n*.

```
(define rad (lambda (k n) (max-rad k n 0 1)))

(define max-rad
  (lambda (k n y p)
    (if (> p n)
        y
        (______ k n ______ (expt (+ y 2) ____)) )))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme per risolvere il seguente problema. Dato un numero naturale *n*, verificare se *n* è esprimibile come potenza intera di base intera $n = k^k$.

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 conoscere il numero di zeri della funzione (cioè quante volte si annulla) 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:

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

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

Definisci formalmente un programma in Scheme basato sulla ricorsione di coda (approccio iterativo) equivalente alla procedura *g*.

```
(define g
  (lambda (n)     ; n naturale
    (if (< n 2)
        1
        (+ (g (- n 2)) (* n (- n 1)))
        )
    ))
```

I Accertamento del Corso di Programmazione – Soluzioni degli esercizi: A

1. Procedure in Scheme

$(f\ 0) \rightarrow 0$	$(f\ 1) \rightarrow 1$	$(f\ 2) \rightarrow 4$	caso generale:
$(f\ 3) \rightarrow 9$	$(f\ 4) \rightarrow 16$	$(f\ 5) \rightarrow 25$	$(f\ n) \rightarrow n^2$

2. Procedure in Scheme

Definizione completa:

```
(define mcm (lambda (x y) (min-multiplo x x y)))

(define min-multiplo
  (lambda (m x y)
    (if (= _(remainder m y)_ 0)
        m
        ( _min-multiplo_ (+ m x) _x_ _y_ )
    )))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define num-divisori
  (lambda (n)  ; n > 1 naturale
    (num-divisori-da 2 n)
    ))

(define num-divisori-da
  (lambda (d n)
    (cond ((= d n) 0)
          ((= (remainder n d) 0)
           (+ 1 (num-divisori-da (+ d 1) n)))
          (else (num-divisori-da (+ d 1) n))
    )))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define si-annulla?
  (lambda (f a b)   ; f procedura
    (cond ((> a b) #f)
          ((= (f a) 0) #t)
          (else (si-annulla? f (+ a 1) b))
    )))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

$(f\ 1) \rightarrow (1-1) \cdot 2^1$

Ipotesi induttiva: *per n > 1*

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

Proprietà dimostrata nel passo induttivo: *per n > 1*

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

Dimostrazione:

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

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

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

6. Ricorsione di coda

Definizione:

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

(define g-iter
  (lambda (n h)
    (if (< n 2)
        h
        (g-iter (- n 2) (* h (* n n)))
        )
    ))
```

I Accertamento del Corso di Programmazione – Soluzioni degli esercizi: B

1. Procedure in Scheme

(f 0) $\rightarrow$ *0*   *(f 1)* $\rightarrow$ *1*   *(f 2)* $\rightarrow$ *8*

(f 3) $\rightarrow$ *27*   *(f 4)* $\rightarrow$ *64*   *(f 5)* $\rightarrow$ *125*

caso generale:

(f n) $\rightarrow n^3$

2. Procedure in Scheme

Definizione completa:

```
(define mcm (lambda (x y) (min-multiplo x y 1)))

(define min-multiplo
  (lambda (x y k)
    (if (= _(remainder (* k y) x)_ 0)
        (* k y)
        ( _min-multiplo_ _x_ _y_ (+ k 1))
    )))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define max-divisore
  (lambda (n)  ; n > 1 naturale
    (max-divisore-da (- n 1) n)
    ))

(define max-divisore-da
  (lambda (d n)
    (cond ((= d 1) "indefinito")  ; nessuna soluzione
          ((= (remainder n d) 0) d)
          (else (max-divisore-da (- d 1) n))
    )))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define esiste-punto-fisso?
  (lambda (f a b)   ; f procedura
    (cond ((> a b) #f)
          ((= (f a) a) #t)
          (else (esiste-punto-fisso? f (+ a 1) b))
    )))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

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

Ipotesi induttiva: *per n > 1*

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

Proprietà dimostrata nel passo induttivo: *per n > 1*

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

Dimostrazione:

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

$$\rightarrow\ (+\ \ 2+(n{-}3)\cdot 2^{n-1}\ \ (n{-}1)\cdot 2^{n-1})$$

$$\rightarrow 2+(n{-}3)\cdot 2^{n-1}+(n{-}1)\cdot 2^{n-1} = 2+(n{-}2)\cdot 2^n$$

6. Ricorsione di coda

Definizione:

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

(define g-iter
  (lambda (n h)
    (if (< n 2)
        h
        (g-iter (- n 2) (* h (* 2 n)))
        )
    ))
```

I Accertamento del Corso di Programmazione – Soluzioni degli esercizi: C

1. Procedure in Scheme

(f 0) $\rightarrow$ *1*　　*(f 1)* $\rightarrow$ *2*　　*(f 2)* $\rightarrow$ *4*

(f 3) $\rightarrow$ *8*　　*(f 4)* $\rightarrow$ *16*　　*(f 5)* $\rightarrow$ *32*

caso generale:

(f n) $\rightarrow 2^n$

2. Procedure in Scheme

Definizione completa:

```
(define rad (lambda (k n) (max-rad k n 1)))

(define max-rad
  (lambda (k n r)
    (if (> _(expt r k)_ n)
        (- r 1)
        ( _max-rad_ _k_ _n_ (+ r 1))
    )))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define fattoriale?
  (lambda (n)  ; n > 0 naturale
    (tutti-i-divisori-da? 1 n)
    ))

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

4. Definizione di procedure in Scheme

Definizione:

```
(define punto-di-massimo
  (lambda (f a b)  ; f procedura, a <= b
    (if (= a b)
        a
        (let ((candidato (punto-di-massimo f (+ a 1) b)))
          (if (> (f a) (f candidato)) a candidato))
    )))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

$$(f\ 1) \rightarrow \frac{(2+1)(1+1)1}{6}$$

Ipotesi induttiva: *per* $n > 1$

$$(f\ n{-}1) \rightarrow \frac{(2n-1)n(n-1)}{6}$$

Proprietà dimostrata nel passo induttivo: *per* $n > 1$

$$(f\ n) \rightarrow \frac{(2n+1)(n+1)n}{6}$$

Dimostrazione:

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

$$\rightarrow (+\ \frac{(2n-1)n(n-1)}{6}\ \ n^2)$$

$$\rightarrow \frac{(2n-1)n(n-1)}{6} + n^2 = \frac{(2n+1)(n+1)n}{6}$$

6. Ricorsione di coda

Definizione:

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

(define g-iter
  (lambda (n h)
    (if (< n 2)
        h
        (g-iter (- n 2) (+ h (* n (expt 2 n))))
        )
    ))
```

I Accertamento del Corso di Programmazione – Soluzioni degli esercizi: D

1. Procedure in Scheme

(f 0) → 1 *(f 1) → 1* *(f 2) → 1*

(f 3) → 6 *(f 4) → 24* *(f 5) → 120*

caso generale:

(f n) → n!

2. Procedure in Scheme

Definizione completa:

```
(define rad (lambda (k n) (max-rad k n 0 1)))

(define max-rad
  (lambda (k n y p)
    (if (> p n)
        y
        ( _max-rad_ k n _(+ y 1)_ (expt (+ y 2) _k_ ))
    )))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define potenza-k-k?
  (lambda (n)  ; n > 0 naturale
    (tutti-i-casi-da? 1 n)
    ))

(define tutti-i-casi-da?
  (lambda (k n)
    (cond ((> k n) #f)
          ((= (expt k k) n) #t)
          (else (tutti-i-casi-da? (+ k 1) n))
    )))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define num-zeri
  (lambda (f a b)   ; f procedura
    (cond ((> a b) 0)
          ((= (f a) 0) (+ 1 (num-zeri f (+ a 1) b)))
          (else (num-zeri f (+ a 1) b))
          )
    ))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

$$(f\ 1) \rightarrow \frac{1(1-1)}{3}$$

Ipotesi induttiva: *per* $n > 1$

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

Proprietà dimostrata nel passo induttivo: *per* $n > 1$

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

Dimostrazione:

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

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

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

6. Ricorsione di coda

Definizione:

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

(define g-iter
  (lambda (n h)
    (if (< n 2)
        h
        (g-iter (- n 2) (+ h (* n (- n 1))))
        )
    ))
```