Corso di Programmazione

I Accertamento del 3 Dicembre 2002 / A

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando lo spazio aggiuntivo. 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 *s* ?
Calcola i risultati della valutazione delle espressioni Scheme:

(s 0 2) (s 1 2) (s 2 2) (s 3 2)

e ipotizza il risultato della generica valutazione *(s n 2)* per $n \geq 0$.

```
(define s
  (lambda (i j)
    (if (or (= i 0) (= j 1))
        1
        (+ (s i (- j 1))
           (* j (s (- i 1) j)))
    )))
```

2. Procedure in Scheme

Completa il programma in Scheme a lato per verificare se una stringa è composta da sole lettere dell'alfabeto. (Per comodità, si assume che le tutte le lettere siano minuscole e che la stringa vuota sia alfabetica.)

```
(define alphabetic?
  (lambda (s)
    (let ((n  ________ ))
      (if (= n 0) #t
          (let ((c (string-ref s 0)))
            (if (and ________  (char<=? c #\z))
                ( ________  (substring s 1 n))
                 ________
                )) ))
    ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme che, data una parola rappresentata da una stringa, assuma come valore la corrispondente parola con tutte le lettere maiuscole (e lasciando inalterati gli altri eventuali caratteri).

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme che, date due funzioni *f, g* dai naturali ai naturali, assuma come valore la funzione *h* tale che, per ogni *x* naturale, *h(x)* sia il massimo fra *f(x)* e *g(x)*.

5. Dimostrazioni per induzione

Considera la procedura *s* dell'esercizio 1 e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(s 1 k)* è dato da $k\,(k + 1) / 2$ per $k \geq 1$.

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

Utilizzando la ricorsione di coda (approccio iterativo), scrivi un programma in Scheme che, dato un numero naturale positivo *n* e un numero primo *p*, permetta di determinare l'esponente di *p* nella scomposizione di *n* in fattori primi.

Corso di Programmazione

I Accertamento del 3 Dicembre 2002 / B

cognome e nome

Risolvi i seguenti esercizi, riporta le soluzioni in modo chiaro negli appositi riquadri e giustifica sinteticamente le risposte utilizzando lo spazio aggiuntivo. 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 *s* ?
Calcola i risultati della valutazione delle espressioni Scheme:

(s 1 1) (s 1 2) (s 1 3) (s 1 4)

e ipotizza il risultato della generica valutazione *(s 1 n)* per $n \geq 1$.

```
(define s
  (lambda (i j)
    (if (or (= i 0) (= j 1))
        1
        (+ (s i (- j 1))
           (* j (s (- i 1) j)))
    )))
```

2. Procedure in Scheme

Completa il programma in Scheme a lato per verificare se una stringa rappresenta un numero naturale. (Per comodità, si assume che la stringa vuota rappresenti un numero.)

```
(define numeric?
  (lambda (s)
    (let ((n  ________ ))
      (if (= n 0) #t
          (let ((c (string-ref s 0)))
            (if (or (char<? c #\0)  ________ )
                ________
                ( ________  (substring s 1 n))
                )) ))
    ))
```

3. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme che, data una parola rappresentata da una stringa, assuma come valore la corrispondente parola con tutte le lettere minuscole (e lasciando inalterati gli altri eventuali caratteri).

4. Definizione di procedure in Scheme

Definisci formalmente una procedura in Scheme che, date due funzioni *f*, *g* dai naturali ai naturali, assuma come valore la funzione *h* tale che, per ogni *x* naturale, *h(x)* sia il minimo fra *f(x)* e *g(x)*.

5. Dimostrazioni per induzione

Considera la procedura *s* dell'esercizio 1 e dimostra per induzione che il risultato della valutazione dall'espressione Scheme *(s n 2)* è dato da $2^{n+1} - 1$ per $n \geq 0$.

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

Utilizzando la ricorsione di coda (approccio iterativo), scrivi un programma in Scheme che, dato un numero naturale positivo *n*, permetta di determinare il più piccolo divisore di *n* maggiore di 1.

I Accertamento di Programmazione / A

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

(s 0 2) $\rightarrow$ *1* *(s 1 2)* $\rightarrow$ *3*

(s 2 2) $\rightarrow$ *7* *(s 3 2)* $\rightarrow$ *15*

caso generale:

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

2. Procedure in Scheme

Definizione completa:

```
(define alphabetic?
  (lambda (s)
    (let ((n  (string-length s) ))
      (if (= n 0) #t
          (let ((c (string-ref s 0)))
            (if (and (char>=? c #\a)  (char<=? c #\z))
                ( alphabetic?  (substring s 1 n))
                 #f
                )) ))
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define uppercase
  (lambda (word)
    (if (= (string-length word) 0)
        ""
        (string-append
         (string (uc (string-ref word 0)))
         (uppercase (substring word 1 (string-length word)))
         )
        )))

(define uc
  (lambda (c)
    (if (and (char>=? c #\a) (char<=? c #\z))
        (integer->char
         (+ (char->integer #\A)
            (- (char->integer c) (char->integer #\a))) )
        c
        )))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define floor
  (lambda (f g)
    (lambda (x)
      (if (< (f x) (g x)) (f x) (g x))
      )))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

$(s\ 1\ 1) \rightarrow 1 = 1 \cdot (1+1) / 2$

Proprietà dimostrata nel passo induttivo: per $k > 1$

$(s\ 1\ k) \rightarrow k\,(k+1) / 2$

Ipotesi induttiva: per $k > 1$

$(s\ 1\ k-1) \rightarrow (k-1)\,k / 2$

6. Ricorsione di coda

```
(define exponent
  (lambda (n p)
    (exp-iter n p 0)
    ))

(define exp-iter
  (lambda (n p e)
    (if (= (remainder n p) 0)
        (exp-iter (quotient n p) p (+ e 1))
        e
        )))
```

I Accertamento di Programmazione / B

Soluzioni degli esercizi:

cognome e nome

1. Procedure in Scheme

(s 1 1) $\rightarrow$ *1* *(s 1 2)* $\rightarrow$ *3*

(s 1 3) $\rightarrow$ *6* *(s 1 4)* $\rightarrow$ *10*

caso generale:

(s 1 n) $\rightarrow$ *n (n + 1) / 2*

2. Procedure in Scheme

Definizione completa:

```
(define numeric?
  (lambda (s)
    (let ((n  (string-length s) ))
      (if (= n 0) #t
          (let ((c (string-ref s 0)))
            (if (or (char<? c #\0)  (char>? c #\9) )
                #f
                ( numeric?  (substring s 1 n))
                )) ))
    ))
```

3. Definizione di procedure in Scheme

Definizione:

```
(define lowercase
  (lambda (word)
    (if (= (string-length word) 0)
        ""
        (string-append
         (string (lc (string-ref word 0)))
         (lowercase (substring word 1 (string-length word)))
         )
        )))

(define lc
  (lambda (c)
    (if (and (char>=? c #\A) (char<=? c #\Z))
        (integer->char
         (+ (char->integer #\a)
            (- (char->integer c) (char->integer #\A))) )
        c
        )))
```

4. Definizione di procedure in Scheme

Definizione:

```
(define ceiling
  (lambda (f g)
    (lambda (x)
      (if (> (f x) (g x)) (f x) (g x))
      )))
```

5. Dimostrazioni per induzione

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

Proprietà dimostrata nel caso base:

$(s\ 0\ 2) \rightarrow 1 = 2^{0+1} - 1$

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

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

Ipotesi induttiva: per $n > 0$

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

6. Ricorsione di coda

```
(define min-div
  (lambda (n)
    (min-div-iter n 2)
    ))

(define min-div-iter
  (lambda (n d)
    (if (= (remainder n d) 0)
        d
        (min-div-iter n (+ d 1))
        )))
```