Corso di Programmazione

III Accertamento del 19 Giugno 2002 / A

cognome e nome

Risolvi i seguenti esercizi e riporta le soluzioni in modo chiaro su questo foglio, giustificando sinteticamente le risposte.

1. Procedure in Scheme

Definisci una procedura in Scheme che, dato un vettore *v* di *n* numeri interi ordinati in ordine crescente e dato un qualunque numero intero *x*, sostituisce la più piccola componente di *v* con *x* e riordina il vettore "spostando" opportunamente le componenti. Per esempio, se inizialmente *v* contiene le componenti 5, 12, 15, 20, 28 (nell'ordine) e *x* = 21, allora dopo l'esecuzione della procedura *v* deve contenere le componenti 12, 15, 20, 21, 28 (sempre ordinate).

2. Programmi iterativi in Java

Definisci un programma in Java che risolva lo stesso problema proposto dall'esercizio n. 1.

3. Classi in Java

Considera la classe *Registro*, rappresentato da un vettore di *n* bit (valori 0 / 1) e accessibile attraverso i metodi indicati sotto, per simulare alcune delle operazioni con i registri di un calcolatore. In particolare, il costruttore inizializza tutti i bit del registro a 0; *carica* rappresenta attraverso i bit del registro il numero naturale passato come parametro; *copia* riproduce il registro passato come parametro; *somma* e *sottrai* effettuano la corrispondente operazione in base due, utilizzando il registro passato come parametro nel ruolo di secondo operando; *spostaDestra* e *spostaSinistra* determinano lo spostamento dei bit di una posizione e assumono come valore il valore del bit che "esce" a destra o a sinistra (nel verso opposto entra uno 0); infine, *minore* e *uguale* determinano il risultato booleano del corrispondente confronto con il registro passato come parametro.

```
public class Registro {

  private static int n = ...;
  private int[] R = new int[n];  // R[i] può essere solo 0 o 1

  public Registro() { ... }

  public void carica(int val) { ... }  // val ≥ 0
  public void copia(Registro reg) { ... }
  public void somma(Registro reg) { ... }
  public void sottrai(Registro reg) { ... }
  public int spostaDestra() { ... }
  public int spostaSinistra() { ... }
  public boolean minore(Registro reg) { ... }
  public boolean uguale(Registro reg) { ... }
}
```

Definisci in Java i metodi *carica*, *sottrai*, *spostaDestra* e *uguale*.
Quindi definisci un metodo, esterno alla classe *Registro*, per moltiplicare i numeri rappresentati da due registri:

```
public Registro moltiplica(Registro x, Registro y)
```

Non si richiede in ogni caso di gestire gli eventuali errori.

4. Astrazione sui dati

Imposta in Scheme l'equivalente della classe *Registro* definita nell'esercizio n. 3, ipotizzando di rappresentare i registri attraverso vettori di *n* bit utilizzabili come variabili di stato. Definisci quindi le procedure che svolgono funzioni equivalenti a *carica*, *sottrai*, *spostaDestra* e *uguale*.

Corso di Programmazione

III Accertamento del 19 Giugno 2002 / B

cognome e nome

Risolvi i seguenti esercizi e riporta le soluzioni in modo chiaro su questo foglio, giustificando sinteticamente le risposte.

1. Procedure in Scheme

Definisci una procedura in Scheme che, dato un vettore *v* di *n* numeri interi ordinati in ordine decrescente e dato un qualunque numero intero *x*, sostituisce la più piccola componente di *v* con *x* e riordina il vettore "spostando" opportunamente le componenti. Per esempio, se inizialmente *v* contiene le componenti 31, 22, 15, 10, 8 (nell'ordine) e *x* = 25, allora dopo l'esecuzione della procedura *v* deve contenere le componenti 31, 25, 22, 15, 10 (sempre ordinate).

2. Programmi iterativi in Java

Definisci un programma in Java che risolva lo stesso problema proposto dall'esercizio n. 1.

3. Classi in Java

Considera la classe *Registro*, rappresentato da un vettore di *n* bit (valori 0 / 1) e accessibile attraverso i metodi indicati sotto, per simulare alcune delle operazioni con i registri di un calcolatore. In particolare, il costruttore inizializza tutti i bit del registro a 0; *carica* rappresenta attraverso i bit del registro il numero naturale passato come parametro; *copia* riproduce il registro passato come parametro; *somma* e *sottrai* effettuano la corrispondente operazione in base due, utilizzando il registro passato come parametro nel ruolo di secondo operando; *spostaDestra* e *spostaSinistra* determinano lo spostamento dei bit di una posizione e assumono come valore il valore del bit che "esce" a destra o a sinistra (nel verso opposto entra uno 0); infine, *minore* e *uguale* determinano il risultato booleano del corrispondente confronto con il registro passato come parametro.

```
public class Registro {

  private static int n = ...;
  private int[] R = new int[n];  // R[i] può essere solo 0 o 1

  public Registro() { ... }

  public void carica(int val) { ... }  // val ≥ 0
  public void copia(Registro reg) { ... }
  public void somma(Registro reg) { ... }
  public void sottrai(Registro reg) { ... }
  public int spostaDestra() { ... }
  public int spostaSinistra() { ... }
  public boolean minore(Registro reg) { ... }
  public boolean uguale(Registro reg) { ... }
}
```

Definisci in Java il costruttore *Registro()* e i metodi *somma*, *spostaDestra* e *minore*.
Quindi definisci un metodo, esterno alla classe *Registro*, per moltiplicare i numeri rappresentati da due registri:

```
public Registro moltiplica(Registro x, Registro y)
```

Non si richiede in ogni caso di gestire gli eventuali errori.

4. Astrazione sui dati

Imposta in Scheme l'equivalente della classe *Registro* definita nell'esercizio n. 3, ipotizzando di rappresentare i registri attraverso vettori di *n* bit utilizzabili come variabili di stato. Definisci quindi le procedure che svolgono funzioni equivalenti a *Registro()*, *somma*, *spostaDestra* e *minore*.

Corso di Programmazione

III Accertamento del 19 Giugno 2002 / C

cognome e nome

Risolvi i seguenti esercizi e riporta le soluzioni in modo chiaro su questo foglio, giustificando sinteticamente le risposte.

1. Procedure in Scheme

Definisci una procedura in Scheme che, dato un vettore *v* di *n* numeri interi ordinati in ordine crescente e dato un qualunque numero intero *x*, sostituisce la più grande componente di *v* con *x* e riordina il vettore "spostando" opportunamente le componenti. Per esempio, se inizialmente *v* contiene le componenti 5, 12, 15, 20, 28 (nell'ordine) e *x* = 8, allora dopo l'esecuzione della procedura *v* deve contenere le componenti 5, 8, 12, 15, 20 (sempre ordinate).

2. Programmi iterativi in Java

Definisci un programma in Java che risolva lo stesso problema proposto dall'esercizio n. 1.

3. Classi in Java

Considera la classe *Registro*, rappresentato da un vettore di *n* bit (valori 0 / 1) e accessibile attraverso i metodi indicati sotto, per simulare alcune delle operazioni con i registri di un calcolatore. In particolare, il costruttore inizializza tutti i bit del registro a 0; *carica* rappresenta attraverso i bit del registro il numero naturale passato come parametro; *copia* riproduce il registro passato come parametro; *somma* e *sottrai* effettuano la corrispondente operazione in base due, utilizzando il registro passato come parametro nel ruolo di secondo operando; *spostaDestra* e *spostaSinistra* determinano lo spostamento dei bit di una posizione e assumono come valore il valore del bit che "esce" a destra o a sinistra (nel verso opposto entra uno 0); infine, *minore* e *uguale* determinano il risultato booleano del corrispondente confronto con il registro passato come parametro.

```
public class Registro {

  private static int n = ...;
  private int[] R = new int[n];  // R[i] può essere solo 0 o 1

  public Registro() { ... }

  public void carica(int val) { ... }  // val ≥ 0
  public void copia(Registro reg) { ... }
  public void somma(Registro reg) { ... }
  public void sottrai(Registro reg) { ... }
  public int spostaDestra() { ... }
  public int spostaSinistra() { ... }
  public boolean minore(Registro reg) { ... }
  public boolean uguale(Registro reg) { ... }
}
```

Definisci in Java i metodi *copia*, *sottrai*, *spostaSinistra* e *minore*.
Quindi definisci un metodo, esterno alla classe *Registro*, per moltiplicare i numeri rappresentati da due registri:

```
public Registro moltiplica(Registro x, Registro y)
```

Non si richiede in ogni caso di gestire gli eventuali errori.

4. Astrazione sui dati

Imposta in Scheme l'equivalente della classe *Registro* definita nell'esercizio n. 3, ipotizzando di rappresentare i registri attraverso vettori di *n* bit utilizzabili come variabili di stato. Definisci quindi le procedure che svolgono funzioni equivalenti a *copia*, *sottrai*, *spostaSinistra* e *minore*.

Corso di Programmazione

III Accertamento del 19 Giugno 2002 / D

cognome e nome

Risolvi i seguenti esercizi e riporta le soluzioni in modo chiaro su questo foglio, giustificando sinteticamente le risposte.

1. Procedure in Scheme

Definisci una procedura in Scheme che, dato un vettore *v* di *n* numeri interi ordinati in ordine decrescente e dato un qualunque numero intero *x*, sostituisce la più grande componente di *v* con *x* e riordina il vettore "spostando" opportunamente le componenti. Per esempio, se inizialmente *v* contiene le componenti 31, 22, 15, 10, 8 (nell'ordine) e *x* = 12, allora dopo l'esecuzione della procedura *v* deve contenere le componenti 22, 15, 12, 10, 8 (sempre ordinate).

2. Programmi iterativi in Java

Definisci un programma in Java che risolva lo stesso problema proposto dall'esercizio n. 1.

3. Classi in Java

Considera la classe *Registro*, rappresentato da un vettore di *n* bit (valori 0 / 1) e accessibile attraverso i metodi indicati sotto, per simulare alcune delle operazioni con i registri di un calcolatore. In particolare, il costruttore inizializza tutti i bit del registro a 0; *carica* rappresenta attraverso i bit del registro il numero naturale passato come parametro; *copia* riproduce il registro passato come parametro; *somma* e *sottrai* effettuano la corrispondente operazione in base due, utilizzando il registro passato come parametro nel ruolo di secondo operando; *spostaDestra* e *spostaSinistra* determinano lo spostamento dei bit di una posizione e assumono come valore il valore del bit che "esce" a destra o a sinistra (nel verso opposto entra uno 0); infine, *minore* e *uguale* determinano il risultato booleano del corrispondente confronto con il registro passato come parametro.

```
public class Registro {

  private static int n = ...;
  private int[] R = new int[n];  // R[i] può essere solo 0 o 1

  public Registro() { ... }

  public void carica(int val) { ... }  // val ≥ 0
  public void copia(Registro reg) { ... }
  public void somma(Registro reg) { ... }
  public void sottrai(Registro reg) { ... }
  public int spostaDestra() { ... }
  public int spostaSinistra() { ... }
  public boolean minore(Registro reg) { ... }
  public boolean uguale(Registro reg) { ... }
}
```

Definisci in Java i metodi *carica*, *somma*, *spostaSinistra* e *uguale*.
Quindi definisci un metodo, esterno alla classe *Registro*, per moltiplicare i numeri rappresentati da due registri:

```
public Registro moltiplica(Registro x, Registro y)
```

Non si richiede in ogni caso di gestire gli eventuali errori.

4. Astrazione sui dati

Imposta in Scheme l'equivalente della classe *Registro* definita nell'esercizio n. 3, ipotizzando di rappresentare i registri attraverso vettori di *n* bit utilizzabili come variabili di stato. Definisci quindi le procedure che svolgono funzioni equivalenti a *carica*, *somma*, *spostaSinistra* e *uguale*.

Soluzione esercizio 1/A

(Le altre varianti dell'esercizio 1 hanno soluzioni analoghe.)

```
(define rearrange
  (lambda (v n x)  ; n > 0
    (rearrange-iter v n x 0)
    ))

(define rearrange-iter
  (lambda (v n x i)
    (cond ((>= i (- n 1))
           (vector-set! v i x))
          ((<= x (vector-ref v (+ i 1)))
           (vector-set! v i x))
          (else
           (begin
             (vector-set! v i (vector-ref v (+ i 1)))
             (rearrange-iter v n x (+ i 1))
             ))
          )
    ))
```

Soluzione esercizio 2/B

(Le altre varianti dell'esercizio 2 hanno soluzioni analoghe.)

```
public static void rearrange(int[] v, int n, int x) {

  int i = n-1;
  boolean trovato = false;

  while ( !trovato ) {
    if (i <= 0) {
      trovato = true;
    } else if (v[i-1] >= x) {
      trovato = true;
    } else {
      v[i] = v[i-1];
      i = i-1;
    }
  }
  v[i] = x;
}
```

Soluzione esercizio 3

```
public class Registro {

  // Registro:   R[n-1] R[n-2] ... R[1] R[0]
  // a destra il bit meno significativo R[0]

  private static int n = 8;
  private int[] R = new int[n];  // R[i] può essere solo 0 o 1


  public Registro() {

    for (int i=0; i<n; i++) {
      R[i] = 0;
    }
  }


  public void carica(int val) {  // val ≥ 0

    for (int i=0; i<n; i++) {
      R[i] = val % 2;
      val = (int) (val / 2);
    }
  }

  public void copia(Registro reg) {

    for (int i=0; i<n; i++) {
      R[i] = reg.R[i];
    }
  }

  public void somma(Registro reg) {

    int riporto = 0;
    for (int i=0; i<n; i++) {
      R[i] = R[i] + reg.R[i] + riporto;
      if (R[i] > 1) {
        R[i] = R[i] - 2;
        riporto = 1;
      } else {
        riporto = 0;
      }
    }
  }

  public void sottrai(Registro reg) {

    int prestito = 0;
    for (int i=0; i<n; i++) {
      R[i] = R[i] - reg.R[i] - prestito;
      if (R[i] < 0) {
        R[i] = R[i] + 2;
        prestito = 1;
      } else {
        prestito = 0;
      }
    }
  }
```

```
  public int spostaDestra() {

    int esce = R[0];
    for (int i=0; i<n-1; i++) {
      R[i] = R[i+1];
    }
    R[n-1] = 0;
    return esce;
  }

  public int spostaSinistra() {

    int esce = R[n-1];
    for (int i=n-1; i>0; i--) {
      R[i] = R[i-1];
    }
    R[0] = 0;
    return esce;
  }


  public boolean minore(Registro reg) {

    int i = n-1;
    boolean uguali = (R[i] == reg.R[i]);
    while ( uguali & (i > 0) ) {
      i = i - 1;
      uguali = (R[i] == reg.R[i]);
    }
    return (R[i] < reg.R[i]);
  }

  public boolean uguale(Registro reg) {

    int i = n-1;
    boolean uguali = (R[i] == reg.R[i]);
    while ( uguali & (i > 0) ) {
      i = i - 1;
      uguali = (R[i] == reg.R[i]);
    }
    return (R[i] == reg.R[i]);
  }

}
```

Soluzione esercizio 4/D
(Le altre varianti dell'esercizio 4 hanno soluzioni analoghe.)

```
(define carica
  (lambda (r x)
    (carica-iter r x 0)
    ))

(define carica-iter
  (lambda (r x i)
    (vector-set! r i (remainder x 2))
    (if (< (+ i 1) n)
        (carica-iter r (quotient x 2) (+ i 1))
        )
    ))

(define somma
  (lambda (r1 r2)
    (somma-iter r1 r2 0 0)
    ))

(define somma-iter
  (lambda (r1 r2 i riporto)
    (let ((s (+ (vector-ref r1 i) (vector-ref r2 i) riporto)))
      (vector-set! r1 i (if (> s 1) (- s 2) s))
      (if (< (+ i 1) n)
          (somma-iter r1 r2 (+ i 1) (if (> s 1) 1 0))
          )
      )))

(define sposta-sinistra
  (lambda (r)
    (let ((esce (vector-ref r (- n 1))))
      (sposta-iter r (- n 1))
      (vector-set! r 0 0)
      esce
      )))

(define sposta-iter
  (lambda (r i)
    (if (> i 0)
        (begin
          (vector-set! r i (vector-ref r (- i 1)))
          (sposta-iter r (- i 1))
          ))
    ))

(define uguale
  (lambda (r1 r2)
    (uguale-iter r1 r2 0)
    ))

(define uguale-iter
  (lambda (r1 r2 i)
    (cond ((= i n) #t)
          ((not (= (vector-ref r1 i) (vector-ref r2 i))) #f)
          (else (uguale-iter r1 r2 (+ i 1)))
          )
    ))
```