Corso di Programmazione

Esame del 20 Luglio 2006

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Scheme e Java

Traduci la seguente procedura Scheme in un corrispondente metodo in Java, basato sulla stessa struttura ricorsiva.

```
(define count
  (lambda (d n)  ;  0 <= d < 4, n >= 2, n potenza di 2
    (let ((m (quotient n 2))
          (b (if (= d 0) 1 0))
          )
      (if (= n 2)
          b
          (+ (* 2 (count d m))
             (count (remainder (+ d 3) 4) m)
             (count (remainder (+ d 1) 4) m)
             b))
      )))
```

2. Astrazione procedurale

La procedura con valori procedurali *hc* è definita in modo tale che, valutando l'espressione `(f x)` dopo aver associato un valore ad *f* attraverso il costrutto `(define f (hc y))`, si ottiene lo stesso valore dell'espressione `(count x y)`. Coerentemente a quanto indicato nell'esercizio 1, si assume che *x* sia compreso nell'intervallo *[0, 3]* e che *y* sia una potenza di due maggiore dell'unità. Completa il codice Scheme della procedura *hc* riportato qui sotto. Il corpo della procedura può fare riferimento ricorsivo ad *hc*, ma non può chiamare la procedura *count* dell'esercizio 1.

```
(define hc

   .....

   .....

   (let ((m (quotient n 2))
         (b (if (= d 0) 1 0))
         )
     (if (= n 2)
         b

         (+ (* 2  ..... )

              .....

              .....

            b))
     )   ...  )  ; chiusura di tutte le parentesi
```

3. Programmazione dinamica

Trasforma la soluzione dell'esercizio 1 in un programma corrispondente, formalizzato nel linguaggio *Java*, che applichi la tecnica di *programmazione dinamica*, cercando di ridurre quanto più possibile la memoria utilizzata per registrare i valori via via calcolati.

4. Asserzioni e invarianti

Il seguente metodo in Java verifica se un array di interi rappresenta una progressione lineare, cioè se la differenza fra coppie di componenti consecutive (considerate nell'ordine) è costante. Completa il programma introducento opportune asserzioni, specificamente: precondizioni, postcondizioni e invarianti del comando iterativo; proponi inoltre una funzione di terminazione (variante) relativa al ciclo. A tua scelta, puoi formalizzare le asserzioni nel linguaggio *Jass* oppure utilizzando una notazione matematica.

```
public static boolean linear( int[] v ) {

  /**  require  ..........  **/

  int n = v.length;
  int delta = v[1] – v[0];
  int k = 2;
  while ( (k < n) && (v[k] == v[k-1] + delta) )

    /**  invariant  ..........

    ..........

    **/

    /**  variant  ..........  **/ {

    k = k + 1;
  }
  return ( k == n );

  /**  ensure  ..........

  ..........

  **/
}
```

5. Classi in Java

Formalizza in Java una classe *Vect3D* per rappresentare vettori nello spazio tridimensionale. Per ogni oggetto *u* della classe *Vect3D* è definito il seguente protocollo: un costruttore; tre metodi per determinare il valore di ciascuna delle componenti; il metodo *u.length()* che consente di conoscere la lunghezza del vettore; il metodo *u.add(v)* che restituisce la somma vettoriale di *u* e *v*; il metodo *u.rescale(c)* che restituisce un vettore con la stessa direzione di *u*, ma di lunghezza riscalata di un fattore *c*.