Corso di Programmazione

Esame del 16 Settembre 2010

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Definizione di procedure in Scheme

Definisci in Scheme una procedura `monotone-sublists` che, data una lista *s* di interi, restituisce la lista delle sottosequenze di *s* monotone strettamente crescenti, ciascuna rappresentata a sua volta da una lista. Esempi:

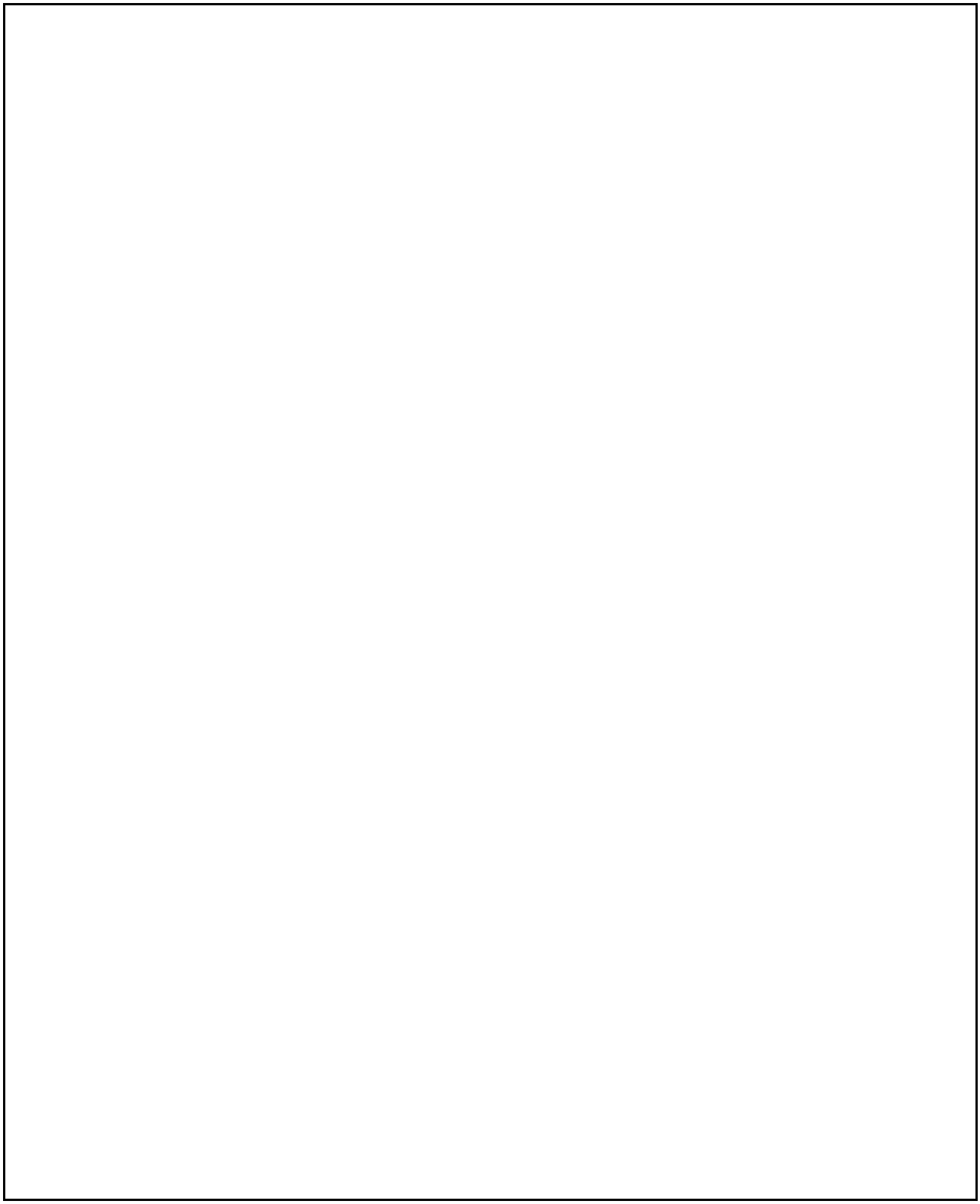
```
(monotone-sublists '())              →  '()
(monotone-sublists '(1))             →  '((1))
(monotone-sublists '(1 2 3 4 5 6))   →  '((1 2 3 4 5 6))
(monotone-sublists '(4 5 6 1 2 3))   →  '((4 5 6) (1 2 3))
(monotone-sublists '(5 6 3 4 1 2))   →  '((5 6) (3 4) (1 2))
(monotone-sublists '(6 5 4 3 2 1))   →  '((6) (5) (4) (3) (2) (1))
(monotone-sublists '(1 2 3 3 2 1))   →  '((1 2 3) (3) (2) (1))
(monotone-sublists '(3 2 1 1 2 3))   →  '((3) (2) (1) (1 2 3))
```

2. Programmazione Dinamica

Considera il seguente metodo statico formalizzato nel linguaggio Java:

```
public static long f( int i, int j, int k ) {  // i, j, k ≥ 0
  if ( i*j*k == 0 ) {
    return  1;
  } else {
    return  f( i-1, j, k ) + f( i, j-1, k ) + f( i, j, k-1 );
}}
```

Trasforma il programma ricorsivo applicando opportunamente la tecnica di *programmazione dinamica*.



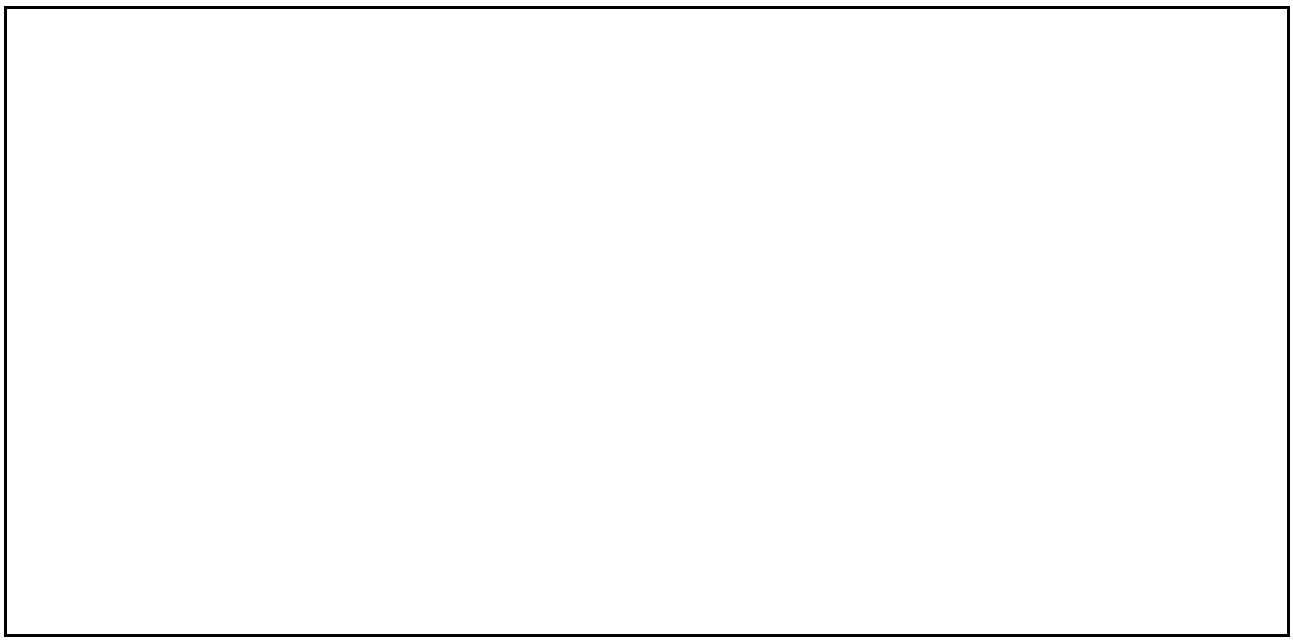
3. Procedure con argomenti e valori procedurali

Formalizza in Scheme una procedura `sumsFromZero` con argomenti e valori procedurali che, data una funzione *f* definita nei numeri naturali e a valori interi, restituisce la funzione *g* tale che, per *x* naturale, *g(x)* è la somma di tutti i valori assunti da *f* nell'intervallo *[0,x]*. Per esempio, sulla base della definizione

```
(define h (sumsFromZero (lambda(x) x)))
```

devono risultare le seguenti valutazioni:

```
(h 0) →  0          (h 1) →  1          (h 2) →  3          (h 3)  →  6
```



4. Correttezza dei programmi iterativi

Dati un array di interi *v* e una coppia di interi *x*, *y*, il seguente metodo statico in Java consente di derminare quanti elementi dell'array ricadono nell'intervallo *[x, y]* (estremi inclusi).
Introduci opportune asserzioni utili a verificare la correttezza del programma, specificamente: *precondizioni*, *postcondizioni* e *invarianti* del comando iterativo. Proponi inoltre una *funzione di terminazione* relativa al ciclo. Non è invece richiesta la dimostrazione formale della correttezza.

```
public static int numberOfValuesWithinSubrange( int[] v, int x, int y ) {

  // Pre:  .....

  int  k = 0,  i = 0,  n = v.length;
  while ( i < n ) {

    // Inv:  .....

    //  .....


    // Term:  .....

    if ( (x <= v[i]) && (v[i] <= y) ) {
      k = k + 1;
    }
    i = i + 1;
  }
  return  k;

  // Post:  .....

  //  .....

}
```

5. Classi in Java

Considera un dispositivo per gestire il pagamento automatico di un parcheggio urbano. La macchina riceve monete di valore non inferiore a 10 *centesimi* e banconote di valore non superiore a 20 *euro*, ma restituisce il resto esclusivamente in monete (10, 20, 50 *centesimi*; 1, 2 *euro*). Per consentire la restituzione del resto in tutti i casi, il dispositivo tiene traccia della propria riserva di monete e si mette "fuori servizio" se il valore complessivo delle monete di riserva è inferiore a 20 *euro* oppure se non ci sono almeno due monete di ciascun tipo.

Definisci in Java una classe `ChangeControl` per modellare il dispositivo descritto sopra. Il protocollo della classe deve prevedere, oltre a un costruttore, opportuni metodi per svolgere le seguenti operazioni:

a. Verificare se la macchina è in servizio;
b. Inserire monete per costituire una riseva iniziale o per accrescerla (metodo con due parametri per definire il numero e il tipo di monete aggiunte);
c. Definire il costo del parcheggio, ovvero stabilire l'importo complessivo da pagare;
d. Introdurre *una* moneta di un certo tipo per pagare (e in tal caso la moneta va ad integrare la riserva);
e. Introdurre *una* banconota di un certo tipo per pagare.

Dopo aver definito il costo, il pagamento viene simulato immaginando di introdurre una moneta o banconota per volta, (punti d, e) in qualsiasi ordine. Non appena l'ammontare complessivo versato supera la somma dovuta, la macchina provvede a calcolare il resto e a restituire (cioè eliminare dalla riserva) le monete necessarie a comporre il resto. Monete o banconote eventualmente inserite prima di definire un (nuovo) costo vengono immediatamente restituite.