Corso di Programmazione

Esame del 2 Settembre 2010

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Procedure in Scheme

Definisci in Scheme una procedura `diffs` che, data una lista non vuota di interi, restituisce la lista delle differenze fra coppie di elementi conscutivi, nell'ordine. (Se l'argomento è una lista di un solo elemento il risultato è la lista vuota.)

Per esempio: `(diffs '(-5 3 7 4 4 5))` → `(8 4 -3 0 1)`

2. Verifica della correttezza

Considera la procedura ricorsiva che hai definito nell'esercizio precedente. Esprimi formalmente le proprietà del risultato restituito in funzione degli argomenti e imposta la dimostrazione per induzione che consente di verificarne la correttezza (è richiesta solo l'impostazione, non lo svolgimento dei passaggi della dimostrzione). Più specificamente:

- Scrivi formalmente la proprietà che intendi dimostrare per induzione:

- Formalizza la proprietà che esprime il caso / i casi base:

(seguito esercizio 2)

- Formalizza l'ipotesi induttiva:

- Formalizza la proprietà da dimostrare come passo induttivo:

3. Procedure con argomenti e valori procedurali

Formalizza in Scheme una procedura `shift` con argomenti e valori procedurali che, data una funzione *f* definita per tutti gli argomenti interi e dato un intero *s*, restituisce la funzione *g* tale che *g(x) = f(x–s)*. Per esempio, sulla base della definizione

```
(define h (shift (lambda(x) (* x x)) 3))
```

devono risultare le seguenti valutazioni:

```
(h 0) → 9        (h 3) → 0        (h 4) → 1        (f 7) → 16
```

4. Memoization

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 *memoization*.

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;
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.