Corso di Programmazione

Esame del 1 Settembre 2003

cognome e nome

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

1. Definizione di procedure in Scheme

Scrivi una procedura *compose* in Scheme che, data una lista $(f_1, f_2, ..., f_{k-1}, f_k)$ di funzioni $f_i : \mathbf{D} \to \mathbf{D}$, definite per tutti gli elementi del dominio **D** e a valori in **D**, assuma come valore la funzione composta $g = f_k \circ f_{k-1} \circ ... \circ f_2 \circ f_1$. In altri termini:

$$(compose \ (list \ f_1 \ f_2 \ ... \ f_{k-1} \ f_k)) \quad \to \quad g$$

dove $g(x) = f_k(f_{k-1}(...f_2(f_1(x))...))$ per ogni $x \in \mathbf{D}$. (Se la lista passata come argomento è vuota, il valore della procedura è la funzione identità, elemento neutro della composizione; presta inoltre attenzione all'ordine delle funzioni nella lista e nella composizione.)

2. Dimostrazioni per induzione

Con riferimento all'esercizio precedente, dimostra per induzione che se tutte le funzioni f_i della lista sono rappresentate dalla stessa procedura (*lambda (x) (* 2 x))* e se *n* è un naturale allora:

$$((compose \ (list \ f_1 \ f_2 \ ... \ f_{k-1} \ f_k)) \ n) \quad \to \quad 2^k \cdot n$$

In particolare:

- Scrivi formalmente la proprietà che intendi dimostrare per induzione.
- 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.

Corso di Programmazione

Esame del 1 Settembre 2003

cognome e nome

Risolvi il seguente esercizio e riporta la soluzione in modo chiaro su questo foglio, giustificandola sinteticamente.

3. Memoization

Applicando una logica analoga alla soluzione del problema della più lunga sottosequenza comune, il seguente metodo in Java valuta il numero di differenze fra due parole *u* e *v*:

```
public static int diff( String u, String v ) {
  int m = u.length();
  int n = v.length();
  if (m == 0) {
    return n;
  } else if (n == 0) {
    return m;
  } else if (u.charAt(0) == v.charAt(0)) {
    return diff( u.substring(1,m), v.substring(1,n) );
  } else {
    return 1 + Math.min( diff(u.substring(1,m),v),
                         diff(u,v.substring(1,n)) );
  }
}
```

Il valore restituito dal metodo è dato dal numero di caratteri di *u* e di *v* che non appartengono a una sottosequenza comune di lunghezza massima, e che quindi non possono essere posti in corrispondenza fra le due parole. La valutazione ricorsiva di *diff(u,v)* è inefficiente perché gli stessi calcoli vengono ripetuti un numero elevato di volte sugli stessi postfissi delle parole *u* e *v*; pertanto è conveniente applicare la tecnica di *memoization*. Definisci un metodo statico in Java basato sulla tecnica di memoization che, date due parole (*String*) *u* e *v*, assuma come valore *diff(u,v)*. Per rappresentare la "storia", tenendo traccia dei calcoli già svolti, utilizza una struttura matriciale di supporto basata sugli *array*.

Corso di Programmazione

Esame del 1 Settembre 2003

cognome e nome

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

4. Classi in Java

La classe *Tree* rappresenta alberi generici non vuoti. Il relativo protocollo è così definito:

```
public class Tree {
  public Tree(int v);
  public Tree(int v, Tree[] subtrees);
  public int rootValue();
  public int numberOfSubtrees();
  public Tree subtree(int k);
}
```

dove il primo costruttore consente di creare un albero di un solo nodo; il secondo un albero con radice di valore *v* e con i sottoalberi dati dagli elementi dell'array *subtrees* (nell'ordine); i metodi *rootValue*, *numberOfSubtrees* e *subtree* restituiscono il valore intero associato alla radice, il numero di sottoalberi (figli della radice) e il *k*-imo sottoalbero, rispettivamente.
Definisci un metodo statico a valori booleani in Java che, dato un valore intero *x* e un albero generico *T* di tipo *Tree*, permetta di verificare se un nodo di valore *x* appartiene all'albero *T*.

5. Astrazione sui dati

Proponi una realizzazione in Scheme degli alberi generici, introdotti nel precedente esercizio, che sia sostanzialmente equivalente alla classe *Tree*. A tale proposito imposta chiaramente il protocollo, elencando le procedure Scheme corrispondenti ai costruttori e ai metodi dell'esercizio precedente, quindi ipotizza un'opportuna rappresentazione degli alberi e definisci il corpo di ciascuna procedura.