Corso di Programmazione

Prova scritta del 5 Settembre 2005

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Astrazione procedurale

Definisci in Scheme una procedura *process* che, dati come parametri una funzione $f : D \times D \rightarrow D$ e una lista non vuota $(x_1\ x_2\ \ldots\ x_n)$ di elementi di D, assuma come valore la lista:

$$\left(x_1 \quad f(x_1,x_2) \quad f(f(x_1,x_2),x_3) \quad f(f(f(x_1,x_2),x_3),x_4) \quad \ldots \quad f(f(\ldots f(f(x_1,x_2),x_3),\ldots,x_{n-1}),x_n) \right) .$$

2. Dimostrazioni per induzione

Con riferimento alla soluzione dell'esercizio precedente, dimostra per induzione che la seguente proprietà vale per ogni n naturale positivo:

$$(process + '(1\ 3\ 5\ 7\ \ldots\ 2n{-}3\ \ 2n{-}1)) \ \rightarrow\ (1\ 4\ 9\ 16\ \ldots\ (n{-}1)^2\ \ n^2)$$

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 occorre dimostrare come passo induttivo.
- Dimostra formalmente il caso base.
- Dimostra formalmente il passo induttivo.

3. Programmazione dinamica

Traduci la seguente procedura Scheme in un corrispondente metodo statico nel linguaggio Java, basato sulla stessa struttura ricorsiva, quindi trasforma il programma, sempre in Java, applicando la tecnica di *programmazione dinamica*.

```
(define fun
  (lambda (n k)  ; 1 <= k <= n
    (if (= n 1)
        k
        (let ((q (+ (* 2 (fun (quotient n 2) 1)) k))
              )
          (if (even? n)
              (+ (remainder (- q 3) n) 1)
              (+ (remainder (- q 1) n) 1)
              ))
        )))
```

4. Astrazione sui dati

Considera la classe *BST* per definire alberi binari di ricerca i cui nodi hanno valori interi. La classe *BST* prevede due costruttori: *BST()* per creare l'albero vuoto e *BST(n,L,R)* per creare l'albero con radice di valore n e con sottoalberi sinistro L e destro R. Sono inoltre disponibili i seguenti metodi: *empty()* per verificare se l'albero è vuoto; *rootNode()* per determinare il valore del nodo radice di un albero non vuoto; *left()*, *right()* per determinare i sottoalberi sinistro e destro di un albero non vuoto. Utilizzando opportunamente il protocollo introdotto sopra, definisci in Java un metodo statico *nodesInInterval(T,x,y)* per calcolare il numero di nodi dell'albero binario di ricerca T il cui valore cade nell'intervallo di interi *[x, y]* (cioè per sapere quanti nodi dell'albero di ricerca cadono nell'intervallo di estremi x, y).

5. Classi in Java

Il gioco del "sudoku" è un rompicapo, recentemente diventato di moda anche in Italia, che si gioca scrivendo cifre comprese fra 1 e 9 nei quadratini di una scacchiera 9x9. L'obiettivo del gioco è compilare tutti i quadratini in modo che ogni riga, ogni colonna e ognuna delle 9 regioni 3x3 (vedi illustrazione a fianco) contenga tutte le cifre da 1 a 9 (oppure equivalentemente: righe, colonne e regioni non devono contenere ripetizioni della stessa cifra). Spesso il gioco viene proposto con alcune cifre già inserite per renderlo più interessante; nella figura è rappresentato un gioco non ancora completo.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4		
4	2	6	8	5				
7	1	3	9					
9	6	1		3		2		
2	8	7						
3	4	5						

Definisci una classe *Sudoku* in Java per realizzare un modello del gioco. Il protocollo deve comprendere un costruttore per creare la scacchiera vuota e due metodi: *disposizioneValida(i,j,c)* per verificare se è possibile inserire la cifra c nel quadratino individuato dalle coordinate di riga i e di colonna j rispettando i vincoli del gioco; *inserisci(i,j,c)* per scrivere la cifra c nel quadratino di coordinate i, j.