Corso di Programmazione

Esame del 24 Settembre 2002

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

Definisci in Scheme una procedura *app-list* che, data una lista di *n* funzioni $f_i : \mathbf{D} \rightarrow \mathbf{D}$ e data una lista di *n* elementi $x_i \in \mathbf{D}$ ($1 \leq i \leq n$), assume come valore la lista di *n* elementi dell'insieme **D** :

$$(f_1(x_1),\ f_2(x_2),\ ...,\ f_n(x_n)) .$$

Quindi, rappresenta il valore della seguente espressione:

(app-list (list (g 1) (g 2) (g 3)) (list 10 20 30))

dove *g* è definito da *(define g (lambda (k) (lambda (n) (+ n k))))* .

2. Dimostrazioni per induzione

In relazione all'esercizio precedente, dimostra per induzione che valutando l'espressione

(app-list (list (g 1) (g 2) ... (g k)) '(1 2 ... k))

si calcola la lista dei primi *k* numeri positivi pari; 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 24 Settembre 2002

cognome e nome

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

3. Classi in Java

Definisci una classe *Sequenze* in Java, che permetta di istanziare sequenze di interi di lunghezza minore o uguale *n*. Tale classe deve prevedere un costruttore per la sequenza vuota e i metodi:

- *open_to_read()* che predispone alla lettura ripartendo dall'inizio della sequenza;
- *read_next()* che legge il successivo elemento;
- *skip_next()* che passa oltre il successivo elemento;
- *end_of_seq()* che verifica se l'intera sequenza è stata percorsa;
- *clear_to_write()* che svuota la sequenza e predispone alla scrittura;
- *write_next(x)* che aggiunge l'intero *x* alla fine della sequenza;
- *clear_last()* che rimuove l'ultimo elemento della sequenza;

In particolare, sono previste due modalità di accesso alle sequenze. La modalità di lettura inizia con *open_to_read* e, attraverso ripetute applicazioni di *read_next* e *skip_next*, consente di accedere agli elementi a partire dal primo secondo l'ordine della sequenza, o semplicemente di saltarli, fino a che *end_of_seq* assume il valore *true*; in ogni caso *open_to_read* riporta all'inizio. La modalità di scrittura inizia con *clear_to_write* che azzera la sequenza (cioè si riparte dalla sequenza vuota) e consente di aggiungere fino ad *n* elementi, nell'ordine, per mezzo di *write_next*, oppure di rimuovere l'ultimo elemento con *clear_last*.

4. Programmi iterativi in Java

Utilizzando istanze della classe definita nell'esercizio precedente, scrivi un frammento di programma iterativo in Java, esterno alla classe *Sequenze*, che duplichi una sequenza data *seq*, cioè che costruisca una copia della sequenza con gli stessi elementi e nello stesso ordine.

5. Programmi su array in Java

Scrivi un metodo statico in Java che, dato un array di interi *A* di lunghezza *n*, restituisca la media aritmetica degli elementi *positivi* di *A*, tralasciando quelli negativi o nulli. Per esempio, se gli elementi dell'array sono *8, -2, 3, 4* e *0,* allora il valore della media è *5*.