Corso di Programmazione

Esame del 14 Settembre 2001

cognome e nome

Risolvi i seguenti esercizi e riporta le soluzioni in modo chiaro nei fogli protocollo, giustificando sinteticamente le soluzioni proposte.

1. Definizione di procedure in Scheme
Scrivi una procedura *op-list* in Scheme che, data una funzione $f : \mathbf{Z}\times\mathbf{Z} \to \mathbf{Z}$, definita per tutte le coppie di numeri interi e a valori interi, e dati tre numeri interi x, y e $n > 0$, assume come valore la lista di lunghezza n così costruita:

$$(x,\ f(x,y),\ f(f(x,y),y),\ f(f(f(x,y),y),y),\ \ldots,\ f(f(f\ldots f(x,y)\ldots,y),y),y))$$

2. Dimostrazioni per induzione
Con riferimento all'esercizio precedente, dimostra per induzione che per ogni $n > 0$:

$$(\textit{op-list}\ \ (\textit{lambda}\ (x\ y)\ (-\ x\ y))\ \ n\ \ 1\ \ n\ ) \ \ \rightarrow \ \ (n\ \ n{-}1\ \ n{-}2\ \ \ldots\ \ 1\ )$$

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 il caso base e il passo induttivo.

3. Classi in Java
Definisci una classe in Java che permetta di istanziare *sequenze* di interi di lunghezza finita. Tale classe deve prevedere un costruttore per rappresentare la sequenza vuota e i seguenti metodi:

- *open_to_read()* che predispone alla lettura ripartendo dall'inizio della sequenza;
- *read()* che legge un elemento e predispone alla lettura del successivo;
- *end_of_seq()* che verifica se tutti gli elementi della sequenza sono stati letti;
- *clear_to_write()* che svuota la sequenza e predispone alla scrittura;
- *write(x)* che aggiunge l'intero x alla fine 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*, consente di accedere agli elementi nell'ordine della sequenza fino a che *end_of_seq* assume il valore *true*; ogni *open_to_read* riporta all'inizio. La modalità di scrittura azzera la sequenza con *clear_to_write* (cioè si riparte dalla sequenza vuota) e consente di aggiungere i vari elementi, nell'ordine, per mezzo di *write*.

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

5. Programmi e invarianti
Completa il programma in Java per calcolare i numeri di Stirling del II tipo in base alle proprietà $\left\{ {i \atop j} \right\} = \left\{ {i-1 \atop j-1} \right\} + j\cdot \left\{ {i-1 \atop j} \right\}$, per $1 < j < i$, e $\left\{ {i \atop 1} \right\} = \left\{ {i \atop i} \right\} = 1$. Dato $n > 0$, $\left\{ {n \atop k} \right\}$ è rappresentato alla fine della computazione dalla componente *s[k]* del vettore *s*, per $k \in [1,n]$. Inoltre, descrivi sinteticamente (in modo informale) quali potrebbero essere le asserzioni invarianti del comando *while* più esterno che dimostrano la correttezza del programma.

```
x = 1;  s[1] = 1;
while  (x < _____ )  {
    y = x;  x = x+1;  s[x] =  _____ ;
    while  (y > 0)  {
        s[y] = s[y-1]+ _____ ;  y =  _____ ;
      }
  }
```