Corso di Programmazione

Esame del 28 Luglio 2010

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Ricorsione di coda

Definisci in Scheme un programma basato su una procedura ricorsiva di coda, più eventuali procedure non ricorsive, per risolvere il seguente problema: data una lista non vuota di interi, determinarne il minimo e il massimo. Il risultato deve essere restituito come lista di due elementi, rispettivamente il minimo e il massimo intero (nell'ordine).

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 valori procedurali

Definisci in Scheme una procedura a valori procedurali `cycle-fun` che, data una lista non vuota *c* di interi, restituisce una funzione *p* periodica di periodo pari alla lunghezza di *c*. Il dominio di *p* è l'insieme dei numeri naturali; a partire dall'argomento 0, *p* assume ciclicamente, e nell'ordine, i valori della lista *c*. Per esempio, sulla base della definizione

```
(define f (cycle-fun '(0 -1 -2 -1 0 1 2 1)))
```

devono risultare le seguenti valutazioni:

```
(f 0) → 0        (f 3) → -1       (f 6) → 2        (f 9) → -1
(f 1) → -1       (f 4) → 0        (f 7) → 1        (f 10) → -2
(f 2) → -2       (f 5) → 1        (f 8) → 0        (f 11) → -1
```

e così via, dove la sequenza di valori si ripete con periodo 8.

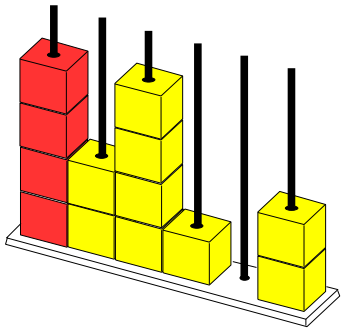
4. Programmazione dinamica

Considera il seguente metodo statico formalizzato nel linguaggio Java:

```
public static long f( int i, int j ) {  // i, j ≥ 0
  if ( i+j < 2 ) {
    return  i+j;
  } else if ( j == 0 ) {
    return  f( 1, i-2 ) + f( 0, i-1 );
  } else if ( j == 1 ) {
    return  f( 0, i ) + f( i+1, 0 );
  } else {
    return  f( i+2, j-2 ) + f( i+1, j-1 );
}}
```

Trasforma il programma ricorsivo applicando opportunamente la tecnica di *programmazione dinamica*.

5. Classi in Java



Considera il gioco per due persone descritto qui di seguito. Una configurazione del gioco è rappresentata da $n+1$ asticelle verticali attraverso le quali possono essere allineati un certo numero di cubetti forati. All'inizio k cubetti rossi sono infilati nell'asticella più a sinistra. Ciascun giocatore, a turno, infila uno o più cubetti gialli in una delle n asticelle inizialmente libere, a sua scelta, eventualmente sopra quelli già presenti, in modo tale che una pila di cubetti gialli non superi mai in altezza la pila di cubetti rossi. Vince il giocatore che "completa la parete", ovvero con la sua (ultima) mossa porta tutte le pile di cubetti alla stessa altezza k. Diverse partite si possono giocare cambiando di volta in volta il numero di asticelle e/o l'altezza della pila iniziale di cubetti rossi.

Definisci in Java una classe `Wall` per modellare il gioco descritto sopra. Il protocollo della classe deve prevedere un costruttore che consenta di istanziare una qualsiasi configurazione iniziale del gioco, nonché opportuni metodi per acquisire informazioni sullo stato del gioco, per verificare se la partita è conclusa e per effettuare le mosse ammissibili.