Corso di Programmazione

Esame del 20 Settembre 2012

| cognome e nome | | |
|----------------|--|--|

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Memoization

Considera il seguente programma ricorsivo in Scheme:

Trasformalo in un programma più efficiente in Java applicando una tecnica top-down di memoization.

2. Programmi in Scheme

Scrivi una procedura is-sorted? in Scheme per verificare se una lista numerica *nums* è ordinata in ordine (non strettamente) crescente. Una lista con meno di due elementi si considera ordinata. Esempi:

```
(is-sorted? '(1/2 1 3/2 1.75 9/5 2 2.5)) → true

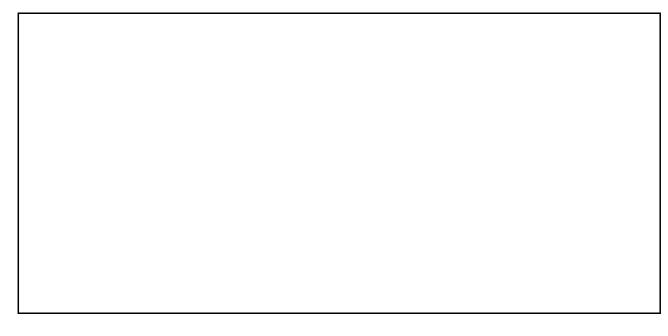
(is-sorted? '(1/2 1 1 3/2 1.75 1.75 9/5 2 2.5)) → true

(is-sorted? '(1/2 1 3/2 9/5 1.75 2 2.5)) → false

(is-sorted? null) → true (is-sorted? '(25)) → true
```

3. Procedure con argomenti e valori procedurali

Definisci in Scheme una procedura diffs che, data una funzione f da interi a interi, restituisce una funzione g da interi a interi con la seguente proprietà: per ogni intero n, g(n) = f(n) - f(n-1). Esempi di applicazione di diffs:



4. Asserzioni e invarianti

Il metodo statico fourthPower è progettato per calcolare la quarta potenza dell'intero passato come argomento *senza* eseguire prodotti. Dimostra fromalmente che l'invariante **Inv**₂ vale all'inizio del secondo ciclo e si conserva.

```
public static int fourthPower( int n ) { // Pre: n \ge 0 int x = 0, y = 0, z = 1, t = n; while ( x < t ) { // Inv<sub>1</sub>: 0 \le x \le n, y = x^2, z = 2x + 1, t = n x = x + 1; y = y + z; z = z + 2; } t = y; while ( x < t ) { // Inv<sub>2</sub>: n \le x \le n^2, y = x^2, z = 2x + 1, t = n^2 x = x + 1; y = y + z; z = z + 2; } return y; // Post: y = n^4
```

| } | |
|---|--|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

5. Classi in Java

Considera il modello della scacchiera realizzato dalla classe Board per affrontare il rompicapo delle n regine. In relazione alla versione discussa a lezione, per le istanze di Board è definito il protocollo richiamato qui sotto:

```
Board( int n ) // costruttore
                                                    int size()
                                                    int queensOn()
void addQueen( int i, int j )
                                                   boolean underAttack( int i, int j )
void removeQueen( int i, int j )
                                                    String arrangement()
```

Ora, senza modificare il protocollo riportato sopra, si vuole raffinare la classe Board in due aspetti:

- (i) Il metodo addQueen non deve permettere di collocare una regina sotto scacco. Se le coordinate i, j individuano una posizione minacciata da altre regine, addQueen deve vuotare la scacchiera riportandola alla configurazione iniziale.
- (ii) Il risultato dell'operazione removeQueen deve essere sempre consistente. Se le coordinate i, j si riferiscono a un quadrato libero (senza regina), similmente a sopra removeQueen deve vuotare completamente la scacchiera.

A tal fine, cambia il tipo della variabile di istanza config, utilizzando un array int[] al posto di String. Gli indici di config sono associati a colonne della scacchiera; il valore di ciascuna componente rappresenta la coordinata della riga in cui è stata collocata una regina nella corrispondente colonna, oppure è zero se non c'è una regina in quella colonna.

Integra la definizione della classe Board in modo da soddisfare i requisiti (i) e (ii) e i vincoli imposti dal protocollo.

| porta solo le modifi rodotte; non riscriv | fiche che si rendono no vere invece le parti che | ecessarie, indica e restano inaltera | ando chiaramente ate rispetto alla vo | in quali punti de ersione discussa a | l codice della class lezione. | se va |
|--|---|---|--|--------------------------------------|----------------------------------|-------|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |