Corso di Programmazione

Esame del 16 Luglio 2012

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Programmi in Scheme

Scrivi un programma in Scheme per risolvere il seguente problema: data una lista *ws* di parole (stringhe) si vuole costruire la lista delle parole che occorrono più di una volta in *ws*. Nella lista risultante ciascuna delle parole ripetute in *ws* deve comparire una volta sola, secondo l'ordine della prima occorrenza in *ws*. Per esempio, a partire dalla lista:

```
("pippo" "pluto" "felix" "yogi" "yogi" "bubu" "braccobaldo" "felix"
                 "topolino" "eta-beta" "bubu" "bubu" "linux" "topolino" "pluto" "felix")
```

si vuole ottenere la lista:

```
("pluto" "felix" "yogi" "bubu" "topolino")
```

2. Programmazione dinamica

Considera il seguente programma ricorsivo in Scheme:

```
(define xpaths
  (lambda (i j u v)               ; i, j, u, v: interi non negativi
    (cond ((and (= i u) (= j v))
           0)
          ((and (= i 0) (= j 0))
           1)
          ((= i 0)
           (xpaths i (- j 1) u v))
          ((= j 0)
           (xpaths (- i 1) j u v))
          (else
           (+ (xpaths (- i 1) j u v) (xpaths i (- j 1) u v)))
          )))
```

Trasformalo in un programma *iterativo* in Java applicando una tecnica bottom-up di *programmazione dinamica*.

3. Procedure con valori procedurali

Data la lista *s* delle cifre, codificate da stringhe di un solo carattere e ordinate per valore crescente, `h` restituisce una procedura per rappresentare gli interi non negativi con le cifre di *s*. Per esempio, la procedura `(h '("0" "1"))` "converte" gli interi nel sistema binario; `(h '("0" "1" "2" "3" "4" "5" "6" "7" "8" "9" "A" "B" "C" "D" "E" "F"))` in quello esadecimale (base 16). Completa accuratamente la definizione della procedura `h`.

```
(define h                   ; valore: procedura (da interi a stringhe)
  (lambda (s)               ; s: lista di stringhe (cifre)

    (let ((b ..................................................... ))
      (lambda (n)           ; n: intero
        (let ((r (remainder n b)) (q (quotient n b)))

          (if ..............................................

              (list-ref s r)

              (string-append ..................... ..................... )
          )))
      )))
```

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* $\boldsymbol{Inv_2}$ vale all'inizio del secondo ciclo e si conserva.

```
public static int fourthPower( int n ) {  // Pre:     n ≥ 0
  int  x = 0,  y = 0,  z = 1,  t = n;
  while ( x < t ) {                       // Inv1:    0 ≤ x ≤ n,  y = x^2,  z = 2x+1,  t = n
    x = x + 1;  y = y + z;  z = z + 2;
  }
  t = y;
  while ( x < t ) {                       // Inv2:    n ≤ x ≤ 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

void addQueen( int i, int j )
void removeQueen( int i, int j )

int size()
int queensOn()
boolean underAttack( int i, int j )
String arrangement()
```

Ora, *senza* modificare il protocollo riportato sopra, si vuole migliorare 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, allora `addQueen` deve lasciare la configurazione della scacchiera inalterata.

(ii) Il risultato dell'operazione `removeQueen` deve essere sempre consistente. Se le coordinate passate a `removeQueen` si riferiscono a un quadrato libero (senza regina), il metodo deve lasciare la configurazione della scacchiera inalterata.

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 alle righe della scacchiera; il valore di ciascuna componente rappresenta la coordinata della colonna in cui è stata collocata una regina nella corrispondente riga, oppure è zero se non c'è una regina in quella riga.

Integra la definizione della classe `Board` in modo da soddisfare i requisiti (i) e (ii) e i vincoli imposti dal protocollo. Riporta solo le modifiche che si rendono necessarie, indicando chiaramente in quali punti del codice della classe vanno introdotte; non riscrivere invece le parti che restano inalterate rispetto alla versione discussa a lezione.