Corso di Programmazione

Esame del 10 Settembre 2018

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Verifica formale della correttezza

In relazione alla procedura `f` riportata qui a fianco, si può dimostrare che per ogni intero $k \geq 0$:

$$(\texttt{f}\ \ 5 \cdot 2^k) \quad \rightarrow \quad 1 + 2^{k+1}$$

Dimostra questa proprietà per induzione sui valori di k attenendoti allo schema delineato qui sotto.

```
(define f
  (lambda (n)                    ; n > 0
    (let ((r (remainder n 4))
          (q (quotient n 4)))
      (if (= q 0)
          (if (= r 2) 1 r)
          (+ (* 4 (f q)) (- (* 2 r) 3))
          ))))
```

- Formalizza la proprietà generale da dimostrare:

- Formalizza la proprietà che esprime il caso / i casi base:

- Formalizza l'ipotesi induttiva:

- Formalizza la proprietà da dimostrare come passo induttivo:

- Dimostra il caso / i casi base:

- Dimostra il passo induttivo:

2. Memoization

Applica la tecnica *top-down* di *memoization* per realizzare una versione più efficiente del seguente programma:

```
public static IntSList lis( int[] s ) {  // s[i] > 0  per  i ∈ [0,n−1]
  int n = s.length;
  return lisRec( s, 0, n, n );
}

private static IntSList lisRec( int[] s, int i, int j, int n ) {
  if ( i == n ) {
    return IntSList.NULL_INTLIST;
  } else {
    IntSList u = lisRec( s, i+1, j, n );
    if ( (j < n) && (s[i] <= s[j]) ) {
      return u;
    } else {
      IntSList v = lisRec( s, i+1, i, n );
      if ( v.length() < u.length() ) { return u; } else { return v.cons(s[i]); }
  }}
}     // la classe  IntSList  modella liste di interi nello stile di Scheme
```

3. Procedure con argomenti e valori procedurali in Scheme

```
(define H
  (lambda (f n)
    (lambda (x y z)
      (if (= y 0)
          z
          ((H f n) (f x x) (quotient y 2)
                   (if (even? y) z (f x z)))
          )) ))
```

```
(define p
  (lambda (x y) ((H + 0) x y 0))
  )

(define q
  (lambda (x y) ((H p 1) x y 1))
  )
```

In relazione al programma riportato sopra, determina il risultato della valutazione delle seguenti espressioni:

`(p 4 0)` → ..... `(p 4 1)` → ..... `(p 4 3)` → .....

`(q 4 0)` → ..... `(q 4 1)` → ..... `(q 4 3)` → .....

`((H append '()) '(1 2 3) 5 '())` → .....

4. Classi in Java

La classe `HanoiPuzzle` consente di modellare il rompicapo della *torre di Hanoi* attraverso il protocollo pubblico:

```
HanoiPuzzle(int n)        // crea la configurazione iniziale con una torre di n dischi nella posizione 0
void move(int s, int d)   // sposta un disco dalla cima della torre in posizione s a quella in d;  s, d ∈{0,1,2}
void show()               // visualizza la configurazione corrente del rompicapo
```

Definisci la classe `HanoiPuzzle` introducendo opportune variabili d'istanza (rappresentazione interna) e realizzando il costruttore e il metodo `move`; non è invece richiesta la realizzazione di `show`.

5. Ricorsione e iterazione

Il seguente programma ricorsivo risolve il rompicapo della torre di Hanoi e "visualizza" la sequenza di configurazioni per una torre iniziale di altezza *n*, utilizzando un'istanza di `HanoiPuzzle` (esercizio **4**) per rappresentarne l'evoluzione.

```
public static void hanoiConfigs( int n ) {
  HanoiPuzzle hanoi = new HanoiPuzzle( n );
  hanoi.show();
  hanoiRec( n, 0, 1, 2, hanoi );
}

private static void hanoiRec( int n, int s, int d, int t, HanoiPuzzle hanoi ) {
  if ( n > 0 ) {
    hanoiRec( n-1, s, t, d, hanoi );
    hanoi.move( s, d );
    hanoi.show();
    hanoiRec( n-1, t, d, s, hanoi );
  }
}
```

Completa la rielaborazione iterativa del programma riportata nel riquadro, dove le quaterne di argomenti delle chiamate ricorsive di `hanoiRec` e le mosse intermedie in sospeso vengono registrate nell'ordine opportuno in uno stack.

```
public static void hanoiConfigsIter( int n ) {

  HanoiPuzzle hanoi = new HanoiPuzzle( n );
  hanoi.show();
  Stack<int[]> stack = new Stack<int[]>();

  stack.push( new int[] { n,  ..... } );
  int s, d, t;

  while ( ..... ) {

    int[] args = ..... ;
    n = args[0];
    s = args[1];  d = args[2];  t = args[3];
    if ( n > 0 ) {

      stack.push( new int[] { ..... } );
      stack.push( new int[] {MOVE,s,d,t} );

      .....

    } else if ( n == MOVE ) {

      hanoi.move( ..... );
      hanoi.show();
  }}
}


private static final int MOVE = -1;
```