Corso di Programmazione

Esame del 25 Luglio 2016

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Astrazione sui dati e dati procedurali

Si vuole introdurre in Scheme un dato astratto *"crivello di Eratostene"* per realizzare un modello del crivello utilizzato nell'algoritmo di Eratostene per determinare tutti i numeri primi che non superano un dato limite *n*. Più specificamente, il protocollo del dato astratto è definito dalle seguenti procedure:

```
(new-sieve)                    ; crea un crivello con potezialmente tutti gli interi ≥ 2
(is-in-sieve p sieve)          ; verifica se p è (ancora) presente nel crivello sieve
(remove-multiples-of p sieve)  ; crivello con il contenuto di sieve tranne i multipli propri di p
(sieve-list sieve n)           ; restituisce la lista dei numeri ≤ n contenuti nel crivello sieve
```

Applicando questa struttura, l'algoritmo per generare i numeri primi non superiori a *n* può essere formulato così:

```
(define primes-list
  (lambda (n)  (eratosthenes 2 n (new-sieve)) ))

(define eratosthenes
  (lambda (p n s)
    (cond ((> p n) (sieve-list s n))
          ((is-in-sieve p s) (eratosthenes (+ p 1) n (remove-multiples-of p s)))
          (else (eratosthenes (+ p 1) n s))
          )))
```

Completa la realizzazione del protocollo in modo tale che siano rispettate le specifiche illustrate sopra.

```
(define new-sieve            ; val: crivello di Eratostene (nuovo)
  (lambda ()                 ; "costruttore" senza argomenti
    (lambda (x) (> x 1))     ; rappresentazione interna procedurale
    ))

(define is-in-sieve          ; val: booleano
  (lambda (p sieve)          ; p: intero,  sieve: crivello di Eratostene
    (sieve p)
    ))

(define remove-multiples-of  ; val: crivello di Eratostene
  (lambda (p sieve)          ; p: intero > 1,  sieve: crivello di Eratostene
    (lambda (x)
      (if ( ..... )
          (= x p)
          .....
          ))
    ))

(define sieve-list           ; val: lista di interi
  (lambda (sieve n)          ; sieve: crivello di Eratostene,  n: intero
    (scan-sieve 2 n sieve)
    ))

(define scan-sieve
  (lambda (x y s)
    (cond ((> x y)  ..... )
          .....
          .....
          .....
          )))
```

2. Programmazione dinamica

Applica la tecnica *bottom-up* di *programmazione dinamica* alla seguente procedura ricorsiva, al fine di trasformarla in un programma iterativo più efficiente.

```
public static long s( int n, int k ) {   // 1 ≤ k ≤ n
  if ( k == n ) {
    return 1;
  } else {
    long x = (n - 1) * s( n-1, k );
    if ( k == 1 ) {
      return x;
    } else {
      return x + s( n-1, k-1 );
  }}
}
```

3. Verifica formale della correttezza

I metodo statico `lcm` calcola il minimo comune multiplo (*mcm*) di due interi positivi *m,n*. Nel programma sono riportate precondizione, postcondizione, invariante e funzione di terminazione. Dimostra la correttezza parziale del programma.

```
public static int lcm( int m, int n ) {

  int x = m;                  // Pre:    m, n > 0
  int y = n;

  while ( x != y ) {          // Inv:    0 < x, y ≤ mcm(m,n),  x % m = 0,  y % n = 0
                              // Term:   2·mcm(m,n) – x – y
    if ( x < y ) {
      int s = y + m – 1;
      x = s - s % m;
    } else {
      int s = x + n – 1;
      y = s - s % n;
    }
  }
  return x;                   // Post:   x = mcm(m,n)
}
```

4. Programmazione in Scheme

Data una lista *ordinata* di numeri con almeno due elementi, la procedura `closest-pair` restituisce una coppia di numeri la cui differenza è minima. Esempio: `(closest-pair '(0.1 0.3 0.5 0.6 0.8 1.1))` → `(0.5 0.6)`. Formalizza un programma in Scheme per `closest-pair`.

5. Ricorsione e iterazione

Dato un *albero di Huffman*, il metodo statico `longestCodeLength` determina la lunghezza del più lungo fra i codici di Huffman associati ai caratteri che compaiono in un documento di testo. Più specificamente, la visita dell'albero, finalizzata alla determinazione di tale lunghezza, è realizzata attraverso uno schema iterativo.
Completa la definizione del metodo `longestCodeLength` riportata nel riquadro.

```
public static int longestCodeLength( Node root ) {

  int lc = ..... ;

  Stack<Node> stack = new Stack<Node>();
  Stack<Integer> depth = new Stack<Integer>();

  stack.push( root );
  depth.push( 0 );

  while ( ..... ) {

    Node n = ..... ;

    int  d = ..... ;

    if ( n.isLeaf() ) {
      lc = Math.max( lc, d );
    } else {

      .....

      .....

      .....

      .....

    }
  }
  return lc;
}
```