Corso di Programmazione

Esame del 21 Gennaio 2019

cognome e nome

1. Procedura in Scheme (vedi esercizio **5**)

```
(define f          ; val: intero
  (lambda (x y)    ; x ≥ 0,  y > 0  interi
    (if (< x y)
        1
        (+ (f (- x 1) y) (f (- x y) y))
        )))
```

2. Ricorsione ad albero

Il programma impostato nel riquadro applica la logica risolutiva del problema della *sottosequenza comune più lunga* (LCS) per confrontare una stringa di riferimento `s` con una diversa versione `t`, dove si assume che sia `s` che `t` siano costituite esclusivamente da lettere maiuscole o minuscole, non da altri tipi di carattere. In particolare, la procedura `xlcs` restituisce una stringa così composta: le lettere di `s` che trovano corrispondenza in `t` sono sostituite da un asterisco `*`; quelle da "cancellare" nella sottosequenza comue più lunga sono rimpiazzate da una barra inclinata `/`; inoltre, le lettere di `t` da "aggiungere" (cioè senza corrispondenza) rispetto alla stringa di riferimento `s` vengono incluse nelle posizioni corrispondenti. Esempi:

```
(xlcs "" "")       →  ""          (xlcs "arto" "atrio")     →  "*/*ri*"
(xlcs "" "ma")     →  "ma"        (xlcs "atrio" "arto")     →  "*/*/t*"
(xlcs "ma" "")     →  "//"        (xlcs "flora" "lira")     →  "/*/i**"
(xlcs "ma" "ma")   →  "**"        (xlcs "cincia" "piani")   →  "/p*a*/*/"
```

Completa il programma inserendo espressioni appropriate negli spazi indicati.

```
(define xlcs                    ; val: stringa
  (lambda (s t)                 ; s, t: stringhe
    (cond ((string=? s "") t)

          (  ..........

           (string-append  ..........  (xlcs (substring s 1) t)))
          ((char=? (string-ref s 0) (string-ref t 0))
           (string-append "*" (xlcs (substring s 1) (substring t 1))))
          (else

           (better (string-append  .......... )

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

(define better
  (lambda (u v)
    (if (< (stars u) (stars v))

        ..........

        ..........
        )))

(define stars
  (lambda (q)
    (if (string=? q "")
        0
        (let ((n (stars (substring q 1))))

          (if (char=?  .......... ) (+ n 1) n)
          ))))
```

3. Programmazione in Scheme

Una stringa *s* è una *palindrome* se procedendo da sinistra a destra oppure da destra a sinistra si legge la stessa sequenza di caratteri, come nel caso di `"esose"`. Scrivi un programma in Scheme per realizzare la procedura a valori interi `palindrome-lev` per determinare il "livello di palindromicità" di una data stringa, cioè il numero di coppie di caratteri uguali in posizioni simmetriche rispetto al centro della stringa (quindi alla stessa distanza rispetto agli estremi), contando anche l'eventuale carattere autosimmetrico che si trovi esattamente al centro della stringa. Esempi:

```
(palindrome-lev "")       →  0     (palindrome-lev "esose")            →  3
(palindrome-lev "a")      →  1     (palindrome-lev "erodere")          →  3
(palindrome-lev "nono")   →  0     (palindrome-lev "ilredevevederli")  →  8
```

(esercizio 3)

4. Memoization

Applica la tecnica *top-down* di memoization e rielabora in Java il programma ricorsivo ad albero che risolve l'esercizio **2**, realizzandone una versione più efficiente.

5. Verifica formale della correttezza

In relazione alla procedura `f` del punto **1**, si può dimostrare che per ogni intero $n \geq 0$:

$$(\texttt{f}\ n\ 1) \quad \rightarrow \quad 2^n$$

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

- 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:

Riporta in modo chiaro negli appositi spazi le soluzioni degli esercizi, oppure precise indicazioni se alcune soluzioni si trovano in un foglio separato. Scrivi inoltre il tuo nome nell'intestazione e su ciascun ulteriore foglio che intendi consegnare.