Programmazione 1 **Esercitazione 2**

Cognome: **Nome:** **Matricola:**

1. Scrivere una procedura che calcoli l'ennesimo numero di Fibonacci usando un **processo iterativo** (si veda il `Lab 4` per la definizione di processo iterativo).

2. Una funzione f è definita dalla regola seguente:

$$f(n) = \begin{cases} n & \text{if } n < 3 \\ f(n-1) + 2\,f(n-2) + 3\,f(n-3) & \text{if } n \geq 3 \end{cases} \tag{1}$$

   (a) Si scriva una procedura che calcoli f usando un **processo ricorsivo**.

   (b) Si scriva una procedura che calcoli f usando un **processo iterativo**.

3. Si scrive un predicato che ci dica se un dato numero n sia un numero primo. Si può usare la definizione che n è un numero primo, se e solo se n è il suo più piccolo divisore maggiore di uno (suggerimento: scrivere prima una funzione che trova il più piccolo divisore di n).

   È possibile scrivere questa procedura in modo tale che l'ordine di crescita per il numero di operazioni richieste sia $\Theta(\sqrt{n})$?

4. La procedura `Sommatoria` vista nel `Lab 7` genera un processo ricorsivo lineare. La stessa procedura può essere riscritta in modo tale che il processo generato sia iterativo: scrivere la procedura che genera un processo iterativo lineare.

5. In maniera analoga alla funzione `Sommatoria`, si può definire una funzione `Produttoria` che restituisce i valori dei prodotti di una funzione valutata in un insieme di punti definiti da un dato intervallo $[a, b]$:

$$\prod_{n=a}^{b} f(n) = f(a) \cdot ... \cdot f(b)$$

Scrivere tale funzione e mostrare come sia possibile usarla per calcolare $n!$

6. Eseguire il grafico sovrapposto delle funzioni seguenti:

$$R(n) = n, R(n) = 10^5\, n, R(n) = n^2, R(n) = \log n, R(n) = n \log n, R(n) = 1.6^n, R(n) = 2^n$$

7. **CHALLENGE (facoltativo)**: Si scriva una procedura che calcoli i numeri di Fibonacci con un ordine di crescita che sia $\Theta(log(n))$. Mandare la soluzione per email.