Programmazione 1 — Tutorato 3

1. Scrivere due funzioni `MapFR(x)` e `FilterFR(x)` che implementano la *map* e la *filter* utilizzando la FoldRight.

```
def FoldRight(P, As, v):
    if IsEmpty(As):
        return v
    return P(Head(As), FoldRight(P, Tail(As), v))
```

2. Scrivere due funzioni `MapFL(x)` e `FilterFL(x)` che implementano la *map* e la *filter* utilizzando la FoldLeft.

```
def FoldLeft(P, As, v):
    if IsEmpty(As):
        return v
    return FoldLeft(P,Tail(As),P(z,Head(As))
```

3. Ricordando le strutture viste:

def $F(X_1, \ldots, X_n)$:
  def $Fiter(Y_1, ..., Y_m)$:
    if $Proposizione(Y_1, ..., Y_m) == True$:
      return $Y_1, ..., Y_{m'}$ $(m' \leq m)$
    return $Fiter(\tilde{Y_1}, ..., \tilde{Y_m})$
  return $Fiter(\tilde{X_1}, ..., \tilde{X_m})$

def $F(X_1, \ldots, X_n)$:
  $Y_i = \tilde{X_i}$ per $i = 0, \ldots, m$
  (le $\tilde{X_i}$ sono costruite a partire dalle X_i)
  while $Proposizione(Y_1, ..., Y_m) == False$:
    $Y_i = \tilde{Y_i}$ per $i = 0, \ldots, m$
  return $Y_1, ..., Y_{m'}$

Si scrivano in entrambi i modi le funzioni seguenti:

(a) `Fib(n)`: prende in input un numero intero n e restituisce l'n-esimo elemento della successione di Fibonacci.

(b) `Fn(n)`: prende in input un numero intero n e restituisce l'n-esimo elemento della successione Fn definita ricorsivamente:

$$\mathrm{Fn}(x) = \begin{cases} n & \text{se } n < 3, \\ f(n-1)f(n-2) + nf(n-3) & \text{se } n \geq 3. \end{cases}$$

(c) `Equal(As,Bs)`: prende in input due liste e restituisce True se sono uguali, False altrimenti.

(d) `PrimiPrimi(n)`: prende in input un numero intero n e restituisce una lista dei primi n numeri primi.

Dando una stima in termini di O-grande della complessità dell'algoritmo.