01. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais),
  nós dizemos que f = Teta(g) (ou f = θ(g)) sse
  existem constantes positivas reais c_1, c_2 e natural positivo n_0 tais que
  0 <= c_1*g(n) <= f(n) <= c_2*g(n) para todo n >= n_0.

02. EXERCÍCIO: mostre que f = Teta(g) sse f = O(g) e f = Ômega(g).

03. OBSERVAÇÕES:

a) A ordenação por inserção executa em tempo O(n) NO MELHOR CASO,
   e em tempo O(n^2) NO PIOR CASO.

b) Observe que a ordenação por inserção executa, no melhor caso,
    também em tempo O(n^2), mas não em tempo Teta(n^2).

c) A ordenação por inserção executa em tempo O(n^2).
   Não é correto, entretanto, dizer que esse algoritmo executa
   em tempo Ômega(n^2).

d) A busca linear executa em tempo O(n) (logo, também em O(n^2), etc).
   A busca linear executa em tempo Ômega(1).
   A busca linear executa, no MELHOR CASO, em tempo O(1) (logo, é também O(lg
n), etc.)
   (Logo, a busca linear não executa em tempo Ômega(n).)
   Logo, a busca linear executa em tempo Teta(1) NO MELHOR CASO.

e) Considere o algoritmo trivial abaixo, que recebe como entrada
   um vetor de n números reais e que retorna o maior dentre
   os dois primeiros elementos do vetor.
   Se t(n) é a função que informa o tempo de execução desse algoritmo,
   então observe que t = O(n). Obviamente, t = O(1) também é verdade,
   e portanto não é verdade que t = Ômega(n), e nem que t = Ômega(lg n), etc.
   Também é verdade que t = Ômega(1), e portanto que t = Teta(1).

```
=======================================================
Algoritmo: maior_dos_2_prim
Entrada: um vetor A[1..n] de números reais, com n >= 2.
Saída: um número real.
-------------------------------------------------------
01. SE A[1] >= A[2]
02. | RETORNE A[1]
03. SENÃO
04. | RETORNE A[2].
=======================================================
```

04. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais),
  nós dizemos que f = o(g) sse para QUALQUER constante real positiva c,
  existe n_0 tal que 0 <= f(n) <= c*g(n) para todo n >= n_0.

05. EXERCÍCIO: mostre que n = o(n^2), e que n^2 != o(n^2).

06. EXERCÍCIO: mostre que, se f = o(g), então f = O(g).

07. EXERCÍCIO: mostre que, se f = O(g), então não necessariamente f = o(g).

08. EXERCÍCIO: mostre que, para qualquer função f, f != o(f).

09. DEFINIÇÃO: Dadas funções f e g de N (Naturais) em R (reais),
  nós dizemos que f = ômega(g) (ou f = ω(g)) sse para QUALQUER constante real
  positiva c, existe n_0 tal que 0 <= c*g(n) < f(n) para todo n >= n_0.

10. EXERCÍCIO: mostre que n^2 = ômega(n).

11. Busca binária recursiva:

```
================================================================
Algoritmo: busca_bin_rec
Entrada: um vetor ORDENADO A[1..n] de números reais, e um real x.
Saída: um natural de 0 a n.
----------------------------------------------------------------
01. RETORNE busca_bin_rec_aux(A, x, 1, n).
================================================================

================================================================
Algoritmo: busca_bin_rec_aux
Entrada: (1) um vetor ORDENADO A[1..n] de números reais,
         (2) um real x,
         (3) um natural i,
         (4) um natural j.
Saída: um natural de 0 a n.
----------------------------------------------------------------
01. SE j < i
02. | RETORNE 0.
03. m := (i+j) div 2.
04. SE x = A[m]
05. | RETORNE m.
06. SE x < A[m]
07. | RETORNE busca_bin_rec_aux(A, x, i, m-1).
08. SENÃO
09. | RETORNE busca_bin_rec_aux(A, x, m+1, j).
================================================================
```

12. A seguinte função limita superiormente o tempo de execução de busca_bin_rec_aux:
    t(1) = c_1.
    t(m) = c_2 + t(y), para algum y <= piso(m/2).

13. EXERCÍCIO: mostre que t(m) <= c_1 + c_2*teto(lg m).