1) Considere as seguintes funções

```
def divisivel(a,b):
    return a%b==0
```

```
def ndiv(n):
    nd=0
    i=1
    while i<=n:           (1)
      if divisivel(n,i):  (2)
         nd=nd+1          (3)
      i=i+1               (4)
    return nd
```

```
def maxdivs(m):
    mx=0
    n=1
    while n<=m:
        if ndiv(n)>mx:
            mx = ndiv(n)
        n=n+1
    return mx
```

```
def primo(n):
    return ndiv(n)==2

def primos_ate(m):
    n=1
    while n<=m:
        if primo(n):
            print n,
        n=n+1
```

a) Descreva, usando no máximo 2 linhas de texto, o efeito de cada uma das funções.
b) Para cada função indique o domínio e o contradomínio, como em (exemplo hipotético): f: N*N -> Bool
   domínio: (tipo dos argumentos) inteiros
   contradomínio: Bool = {False, True}.
c) Mostre que se um inteiro n tem um divisor < sqrt(n), tem outro divisor > sqrt(n) e vice-versa, onde sqrt(n) representa a raíz quadrada de n.
d) Com base no resultado da alínea anterior, implemente uma versão (muito) mais eficiente de "ndiv", com nome "ndiv1", em que só se testa a divisibilidade por inteiros (positivos) <= sqrt(n). Não use "**" nem a função "math.sqrt".
e) Teste (mentalmente) o bom funcionamento para n = 2, 6, 25.
f) Com base no número de vezes que a linha (2) é executada, e supondo que cada execução corresponde a 1 microsegundo do tempo total, compare o tempo de execução de "ndiv(10000000)" e "ndiv1(10000000)".

2) Com base no que aprendeu no problema anterior, implemente uma
função mais eficiente para "primo(n)".
A sua implementação não deve usar funções auxiliares (com a
possível excepção de "divisivel"); a detecção da eventual
não-primalidade de n (e consequente retorno da função) deve ocorrer
o mais cedo possível.

---

3) Seja n um inteiro positivo.
a) Implemente

i) de forma iterativa - com um ciclo, mas sem chamadas a
funções
ii) de forma recursiva - sem ciclos, mas podendo chamar a
própria função

uma função

ndig(n): N -> N

que, com base em divisões inteiras por 10, calcule o número de
dígitos de n.
Nota: em python há outros métodos que se podem utilizar para o fim
pretendido; por exemplo, ndiv(n)=len(str(n)). Mas neste exercício
não pode usar funções "extra" como "str" nem o operador "**".

b) Determine uma fórmula matemática fechada para ndig(n). Por
exemplo, deverá ser ndig(9980)=4.