Introdução à Programação (CC,ERSI) 2009/2010 – exame, 2ª parte

1. **Seja `li` uma lista de inteiros e `x` um inteiro. Escreva uma função `quantas(x,li)` que retorna o número de ocorrências de `x` na lista `li`. Exemplos:**

   `quantas(2,[3,2,1,9,2,2,0,0,2,11,-1,2])` → **5**
   `quantas(5,[50])` → **0**

```
def quantas(x,li):
    c=0
    for y in li:
        if x==y:
            c=c+1
    return c
```

2. **Escreva uma função `h(d)` que retorna o menor inteiro (positivo) que tem pelo menos `d` divisores (`d` é um inteiro positivo).**
   Exemplo: `h(4)`→6, uma vez que 6 tem 4 divisores e todos os inteiros positivos menores que 6 têm menos que 4 divisores. Sugere-se escrever uma função auxiliar que calcula o número de divisores de um inteiro.

```
def h(d):
    n=1
    while ndivs(n)<d: # quando encontrarmos um inteiro com >= d divisores...
        n=n+1
    return n          # ...retornamos esse inteiro

def ndivs(n):
    c=0
    for d in range(1,n+1):
        if n%d==0:
            c=c+1
    return c
```

3. **Seja `n` um inteiro não negativo e `s` uma string contendo apenas letras minúsculas, todas distintas. Escreva uma função `pals(s,n)` que retorna a lista de todas as strings de comprimento `n` que se podem formar com as letras de `s`. Exemplo:**
   `pals("ab",3)` → `["aaa", "aab", "aba", "abb", "baa", "bab", "bba", "bbb"]`

```
def pals(n,s):      # Usamos indução em 'n'
    if n==0:        # Caso base:
        return [""] # há uma só palavra para n=0: ""
    r=[]
    p=pals(n-1,s)   # para n>=1: as palavras de comprimento n obtêm-se...
    for c in s:     # ...juntando uma das letras possíveis...
        for w in p: # ...a cada palavra de comprimento n-1
            r.append(c+w)
    return r
```

Outra solução. Apresentamos uma solução não recursiva baseada em contar na base `len(s)`, usando como "dígitos" as letras da string `s`; por exemplo:

```
def pals(n,s):
    inicial=[0]*n
    final  =[1]*n
    r=[]
    while inicial!=final:
        r.append(string(inicial,s))
        increm(inicial,len(s))
    return r

def string(li,s): # Ex: string([0,0,1],"ab") -> "aab"
    r=""
    for i in range(len(li)):
        r=r+s[li[i]]
    return r

def increm(li,b):
    """ incrementa o inteiro representado por 'li' (base 'b');
        Ex: li=[0,1,1], increm(li,2) -> li: [1, 0, 0]
    """
    i=len(li)-1
    while i>=0 and li[i]==1:
        li[i]=0
        i=i-1
    if i<len(li):
        li[i]=1
```