Folha prática 11 Introdução à Programação, DCC/FCUP, 2009/2010

1. *Um problema fácil...*
   Escreva uma função `num_digs(n)` que retorna o número de dígitos da representação do inteiro não negativo `n` na base 10. Exemplos

   ```
   num_digs(5)     -> 1
   num_digs(5000)  -> 4
   num_digs(0)     -> 1
   ```

   Generalize a função para o número de dígitos da representação de `n` na base `b`; para isso, escreva a função `num_digsb(n,b)`.

2. *Sequências crescentes*
   Considere a seguinte lista de inteiros

   $$[\underline{1},5,\underline{4},\underline{2,4,8},\underline{5,7}]$$

   As sub-sequências crescentes estão sublinhadas: `(1,5)`, `(4)`, `(2,4,8)` e `(5,7)`. A maior delas tem comprimento 3. Escreva uma função `msc(a)` que calcula o comprimento da maior sub-sequencia crescente da lista de inteiros `a`. Outros exemplos

   ```
   [5,4,3,2,1]          ->  1
   [3,4,2,1,5,6,8,9]    ->  5
   ```

3. *Débitos e créditos*
   Suponha que `deb` é uma lista de débitos (gastos) de uma determinada pessoa, sendo cada elemento da lista (a que se chame *movimento*) um tuplo da forma

   `(ano,mes,dia,valor)`

   Cada movimento é sempre mais recente (ou tem a mesma data) que os movimentos que o precedem (os movimentos mais recentes estão no fim da lista). Existe também uma lista análoga de créditos, `cred`.
   Escreva as seguintes funções:

   (a) `saldo(deb,cred)` que determina o saldo actual,

   (soma de todos os créditos) − (soma de todos os débitos)

   (b) `mgastos(deb,ano,mes)` que calcula o gasto total no mês `mes` do ano `ano`.

   (c) `movimento(lista,ano,mes,dia,valor)` que tem como efeito colocar o tuplo `(ano,mes,dia,valor)` no fim da lista `lista`; esta "função" não retorna qualquer valor. Note que uma chamada como

   `movimento(deb,2009,12,15,100.50)`

   corresponde a um débito, enquanto uma chamada como

`movimento(cred,2009,12,22,1000.99)`

corresponde a um crédito.
Se `(ano,mes,dia)` não for uma data legal[1] ou for uma data mais antiga que a data do último movimento, a `lista` não deve ser alterada e deve ser impresso "data ilegal" (claro que nada se deve imprimir se a data for "legal").

Experimente as funções que implementou com as listas

```
deb  = [(2009,11,10,100.11),(2009,11,21,10.10),(2009,12,1,15.11)]
cred = [(2009,10,25,1500),(2009,11,25,1500)]
```

Pode usar a seguinte função para imprimir uma lista de débitos ou créditos de forma mais agradável.

```
def showtab(a):
    print
    for m in a:
        ano   = m[0]
        mes   = m[1]
        dia   = m[2]
        valor = m[3]
        print "  %4d/%2d/%2d ... %10.2f" % (ano,mes,dia,valor)
```

4. *Determinante de uma matriz*
Se uma matriz quadrada estiver "triangularizada" (todos os elementos abaixo da diagonal principal são nulos) o seu determinante é o produto de todos os elementos dessa diagonal. Por exemplo

$$\det\left(\begin{bmatrix} 2 & 5 & 9 \\ 0 & 4 & 1 \\ 0 & 0 & -1 \end{bmatrix}\right) = 2 \times 4 \times (-1) = -8$$

Escreva uma função `determinante(m)` que calcula o determinante da matriz representada pela lista de listas `m`.

Em primeiro lugar deve "triangularizar" a matriz, o que se consegue multiplicando linhas por constantes apropriadas e somando a outras. No exemplo seguinte

$$\begin{bmatrix} 1 & 4 & -1 \\ 2 & 5 & 1 \\ 1 & 6 & -2 \end{bmatrix} \rightarrow (a) \rightarrow \begin{bmatrix} 1 & 4 & -1 \\ 0 & -3 & 3 \\ 0 & 2 & -1 \end{bmatrix} \rightarrow (b) \rightarrow \begin{bmatrix} 1 & 4 & -1 \\ 0 & -3 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

sendo o determinante $m[0][0] \times m[1][1] \times m[2][2] = -3$. As operações foram

(a) A primeira linha multiplicada por -2 foi adicionada à segunda linha. A primeira linha multiplicada por -1 foi adicionada à terceira linha. Note-se que neste passo a primeira linha não é alterada.

(b) A segunda linha multiplicada por $2/3$ foi adicionada à terceira linha. Note-se que neste passo a segunda linha não é alterada.

[1] Suponha que o mês de Fevereiro tem sempre 28 dias.

No caso geral

```
Para i=0,1,2,...,n-2:
  Para j=i+1,i+2,...,n-1:
    A linha i é multiplicada por -a[j][i]/a[i][i]
      e adicionada à linha j (A linha i não é alterada)
```

Deve usar uma função auxiliar `somalinha(m,i,j,x)` que tem como efeito somar à linha j a linha i multiplicada por x.
Notas. (i) Este algoritmo pode ser estendido por forma a implementar o método de resolução de sistemas de equações conhecido como "método de eliminação de Gauss". (ii) Admite-se que durante o cálculo o valor de `a[i][i]` nunca é nulo; existem técnicas para tratar esse caso.