Segundo Exercício Escolar de Programação 1
(IF968)

Fernando Castor
Centro de Informática
Universidade Federal de Pernambuco

30 de novembro de 2016

1. **(7,0 ptos.)** Um tokenizador é um programa que, dados um string a ser tokenizado e um string contendo caracteres separadores, quebra o primeiro string em palavras com base nos separadores, devolvendo uma lista de palavras (strings) como resultado. Construa uma função chamada `tokenize` que funciona como um tokenizador. Na resolução desta questão, é proibido usar laços (`while`, `for`). Também é proibido usar a função `split` e o operador `in`. Tenha em mente que strings em Python são imutáveis. Logo, não dá para usar a função `pop`. Por outro lado,

A resolução deve estar organizada da seguinte maneira:

(a) (1,0 pto.) uma função para determinar se um string contendo apenas um caractere é um elemento de um segundo string.

```
>>> elem('d', 'abcd')
True

>>> elem('x', 'abcd')
False
```

(b) (2,0 ptos.) uma função para, dados um string de texto e um string contendo separadores, obter a primeira palavra que aparece no primeiro string, por exemplo:

```
>>> pegaPalavra("abc,def,ghi", ",")
'abc'

>>> pegaPalavra("Saitama foi até a lua, mas voltou! Sensacional.", ",! ")
'Saitama'

>>> pegaPalavra("(81)123-456-789", "()-")
'81'
```

No segundo exemplo acima, note que o caractere espaço em branco também aparece como um dos separadores (segundo parâmetro passado para `tokenize`).

(c) (1,0 pto.) uma função para, dados um string de texto e um string contendo separadores, obter toda a parte desse primeiro string que aparece após a primeira palavra. Para o primeiro exemplo do item anterior, essa função devolveria `",def,ghi"`.

(d) (1,0 pto.) uma função para, dados um string de texto e um string contendo separadores, obter toda a parte desse primeiro string que aparece após a primeira sequência de separadores, por exemplo:

```
>>> tiraSeparadores("(81)123-456-789", "()-")
'81)123-456-789'

>>> tiraSeparadores("      ok, meu!", " ")
'ok, meu!'
```

(e) (2,0 ptos.) a função **tokenize** propriamente dita, como o exemplo a seguir:

```
>>> tokenize("abc,def,ghi", ",")
['abc', 'def', 'ghi']

>>> tokenize("Saitama foi até a lua, mas voltou! Sensacional.", ",! ")
['Saitama', 'foi', 'até', 'a', 'lua', 'mas', 'voltou', 'Sensacional']

>>> tokenize("(81)123-456-789", "()-")
['81', '123', '456', '789']
```

2. **(3,0 ptos.)Escolha uma das duas questões a seguir para fazer. Deixe claro em sua prova qual opção você escolheu.** É proibido usar laços (`for`, `while`) e o operador `in` nessas questões.

Opção 1: Construa uma função `sort` que, dada uma lista de números, ordena os elementos dessa lista (em ordem crescente). É proibido usar **o algoritmo quicksort** para resolver esta questão!

Opção 2: Construa uma função que, dadas duas matrizes representadas como listas de listas e com as mesmas dimensões (mesmas quantidades de linhas e colunas), realiza a a soma dessa duas matrizes. Por exemplo:

```
>>> somaMatrizes([[1, 2],
                  [1, 0],
                  [1, 2]],
                 [[0, 0],
                  [7, 5],
                  [2, 1]])
[[1, 2],
 [8, 5],
 [3, 3]]
```

No exemplo acima, as listas de listas foram organizadas de modo a mais claramente representar duas matrizes com duas colunas e três linhas cada.