Lista de Exercícios de Linguagens de Programação II
Universidade Federal do Amazonas
Departamento de Ciência da Computação
Marco Cristo

Funcional

1) O que são funções de primeira ordem?

   Funções que podem ser passadas como parâmetros ou retornadas de outras funções.

2) O que são funções de alta ordem?

   Funções que passam funções como parâmetros e/ou retornam funções.

3) Quais as três funções de alta ordem mais comumente encontradas em Linguagens Funcionais? Para que servem?

   Map (mapear uma função para um conjunto de valores), reduce (reduzir um conjunto de valores para um valor de acordo com uma função) e filter (eliminar de um conjunto elementos que atendem a um critério definido por uma função).

4) Quais as características mais importantes do paradigma funcional?

   Todas as funções são de primeira ordem. Usa funções de alta ordem. Ordem não é importante. Assinalamentos não são modificados. Evitam efeitos colaterais. Usam recursão no lugar de laços. Tradicionalmente, manipulam principalmente uma única estrutura de dados, a lista.

5) Resolva os seguintes problemas em Python, usando o paradigma funcional (note que não é permitido o uso de métodos da classe *list*, nem instruções de iteração):

   a) Escreva uma função que calcule as raízes reais (e complexas) de uma equação quadrática. Os parâmetros para a função são os três coeficientes da equação. Dica: use a função *sqrt*() do pacote *cmath*.

```
from cmath import sqrt
raizes = lambda a,b,c: ((-b - sqrt(b * b - 4 * a * c)) / 2 * a, (-b + sqrt(b * b - 4 * a
* c)) / 2 * a)
```

   b) Escreva uma função recursiva que, dados dois parâmetros numéricos, x e y, retorne x^y.

```
potencia = lambda x,y: 1 if y == 0 else x * potencia(x, y - 1)
```

   c) Escreva uma função que retorne o número de zeros em uma lista.

```
nzeros = lambda l: len([x for x in l if x == 0])
```

   d) Escreva uma função que retorne o maior e o menor número de uma lista.

```
maxmin = lambda l: (reduce(lambda x,y: x if x>y else y, l), reduce(lambda x,y: x if x<y
else y, l))
```

   e) Escreva uma função que remova o último elemento de uma lista.

```
dellast = lambda l: [v for idx,v in enumerate(l) if idx < len(l)-1]
```

f) Escreva uma função que, dada uma lista *l* e um inteiro *n*, é retornada uma lista sem o *n*-ésimo elemento de *l*. Se *n* é maior que o tamanho de *l*, *l* não é modificada. Por exemplo, removaNesimo([1,2,3,4,5,6], 3) → [1,2,4,5,6], removaNesimo([1,2,3,4,5], 42) → [1,2,3,4,5], removaNesimo([2,[1,2],5,[1,[7,[8,2],9],1], 3) → [2,5,[1,[7,[8,2],9],1]

```
deln = lambda l, n: [v for idx,v in enumerate(l) if idx != n - 1]
```

g) Escreva uma função que verifique se duas listas numéricas têm a mesma estrutura. Duas listas têm a mesma estrutura se cada um dos seus elementos tem a mesma estrutura, não importando os valores numéricos. Por exemplo, as listas [1, [2, [3]], 4] e [5, [6, [7]], 8] enquanto [1, 2, 3, [4]] e [[1], 2, [3, 4]] têm estruturas distintas.

```
mesmaestrutura = lambda l1, l2: True if (l1 == [] and l2 == []) else False if (l1 == []
or l2 == []) else mesmaestrutura(l1[1:],l2[1:]) if ((type(l1[0]) is list and type(l2[0])
is list) or (type(l1[0]) is int and type(l2[0]) is int)) else False
```

h) Escreva uma função que, dada uma lista L, é retornada a lista L em uma forma plana (sem sub-listas). Ou seja, elementos de uma sub-lista se tornam elementos da lista de nível mais alto. Por exemplo, nivelado([2,[1,2],5,[1,[7,[8,2]]]]) → [2,1,2,5,1,7,8,2], nivelado([1,2,3,4]) → [1,2,3,4].

```
plana = lambda l: [] if l==[] else [l[0]] + plana(l[1:]) if not (type(l[0]) is list)
else plana(l[0]) + plana(l[1:])
```

i) Escreva uma função que, dadas duas listas ordenadas de inteiros L1 e L2, retorne a concatenação de L1 e L2, preservando a ordem dos elementos na lista resultante. Por exemplo, concatene([1,3,4,5,7,10],[2,4,6,8,9]) → [1,2,3,4,4,5,6,7,8,9,10]; concatene ([1,2],[]) → [1,2]; concatene ([],[1,2]) -> [1,2].

```
merge = lambda l1, l2: l1 if l2 == [] else l2 if l1 == [] else [l1[0]] + merge(l1[1:],
l2) if l1[0] < l2[0] else [l2[0]] + merge(l1, l2[1:])
```