Computação 1, 2021.1
Lista 7

Data limite para entrega: 9 de setembro às 23:59

Submeta suas soluções colocando os arquivos correspondentes na sua pasta do Google Drive*

Atualizada em 3/9, esclarecendo o enunciado da Q2a.

Parte 1 — Obrigatória

Questão 1 (Repetidores)**.** Dada uma função f que pode ser composta consigo própria (i.e., os valores retornados por f também são aceitos como entradas para a própria f) e dado um número natural n, definimos a *n-ésima repetição* de f como sendo a função composta

$$f^n = \underbrace{f \circ (f \circ (f \circ (\cdots \circ (f \circ f) \cdots)))}_{f \text{ aparece } n \text{ vezes}}.$$

(Definimos f^0 como a função identidade.) Faça uma função que receba como entrada f e n e retorne a função f^n.

Questão 2 (Somas parciais)**.** Definimos a "soma parcial" de uma função $f : \mathbb{N} \to \mathbb{R}$ como sendo a função $F : \mathbb{N} \to \mathbb{R}$ tal que

$$F(n) = \sum_{i=0}^{n-1} f(i).$$

a. Escreva uma função que implementa a operação de soma parcial.

b. É possível compor a sua função da letra (a) com ela mesma? Por quê?

Questão 3 (Acumuladores)**.**

a. Faça uma (única) função (de alta ordem) que, quando chamada com os parâmetros apropriados, retorne cada uma das funções S, P, M abaixo. Em cada caso, indique quais são os parâmetros apropriados (ou seja: na letra **a** você vai definir a sua função de alta ordem, e em cada um dos itens de **b** a **d** você vai dizer quais os argumentos que quando passados à função da letra **a** resultam na função pedida naquele item).

*Link recebido por email em 19/7/2021 — o nome é parecido com `<seu nome> - Computação 1 2021.1 - Submissões e Feedback.`

b. $S(i_0, n, f) = \sum_{i=i_0}^{n} f(i)$

c. $P(i_0, n, f) = \prod_{i=i_0}^{n} f(i)$

d. $M(i_0, n, f) = \operatorname*{mdc}_{i=i_0}^{n} f(i) = \text{mdc}(f(i_0), f(i_0+1), \ldots, f(n))$,

sendo o *mdc* de números naturais $n_0, n_1, \ldots, n_k$ o maior número natural que divide *todos* os números $n_0, n_1, \ldots, n_k$, conforme vimos em uma lista anterior. Você pode usar o seguinte fato:

$$\text{mdc}(n_0, n_1, \ldots, n_k) = \text{mdc}(\text{mdc}(\cdots \text{mdc}(\text{mdc}(n_0, n_1), n_2), \cdots), n_{k-1}), n_k),$$

e também pode usar a função `gcd` do módulo `math`, que calcula o mdc de **dois** números.

Questão 4 (Aproximações)**.**

a. Faça uma (única) função (de alta ordem) que, quando chamada com os parâmetros apropriados, retorne uma estimativa para cada uma das constantes abaixo, usando as fórmulas infinitas dadas. Em cada caso, indique quais são os parâmetros apropriados. O valor estimado deve ser retornado quando a diferença (em valor absoluto) entre estimativas consecutivas for menor do que uma certa tolerância, também fornecida como parâmetro para a sua função. Você pode usar a função `sqrt` do módulo `math`, mas não pode usar mais nenhum componente deste módulo (ou seja: na letra **a** você vai definir a sua função de alta ordem, e em cada um dos itens de **b** a **f** você vai dizer quais os argumentos que quando passados à função da letra **a** resultam na aproximação à constante pedida naquele item. Cuidado ao testar suas funções: em alguns casos, a convergência é **muito** lenta, o que significa que tolerâncias muito pequenas podem causar problemas pro seu computador. Não seja muito ambicioso!)

b. $\pi = 2 \cdot \left[\left(2 \cdot \frac{2}{3}\right) \cdot \left(\frac{4}{3} \cdot \frac{4}{5}\right) \cdot \left(\frac{6}{5} \cdot \frac{6}{7}\right) \cdot \left(\frac{8}{7} \cdot \frac{8}{9}\right) \cdots \right]$

c. $\ln(2) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \cdots$

d. $\ln(3) = 1 + \frac{1}{3 \cdot 4} + \frac{1}{5 \cdot 16} + \frac{1}{7 \cdot 64} + \frac{1}{9 \cdot 256} + \frac{1}{11 \cdot 1024} + \cdots$

e. $e = 2 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \frac{1}{720} + \cdots$

f. $\pi = 2 \cdot \left[\frac{2}{\sqrt{2}} \cdot \frac{2}{\sqrt{2+\sqrt{2}}} \cdot \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}} \cdot \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}} \cdots \right]$

Questão 5.

a (Currying)**.** Faça uma função que receba como argumento uma função `f`, que assumimos por hipótese ser uma função que aceita exatamente dois argumentos, e retorne a versão *curryada* desta função, isto é, o retorno deve ser uma função de 1 argumento que, ao receber `x`, retorna uma função de 1 argumento que, ao receber `y`, retorna o valor da função `f` nas entradas `x`, `y`.

b (Uncurrying)**.** Faça uma função que receba como argumento uma função `g`, que assumimos por hipótese ser uma função que aceita exatamente 1 argumento e que retorna uma função de exatamente 1 argumento também, e retorne a versão *uncurryada* desta função, isto é, o retorno deve ser uma função de 2 argumentos que, ao receber `x` e `y`, retorna o valor de `h` calculada na entrada `y`, sendo `h` a função retornada por `g` na entrada `x`.

Parte 2 — Desafio opcional

Questão 6. Como vimos, em `Python` podemos definir funções anônimas usando a palavra-chave `lambda`. A inspiração para esta notação é um sistema lógico formal chamado *Cálculo Lambda*, inventado por Alonzo Church na década de 1930 como uma forma de se estudar a noção de *computabilidade* de funções entre números naturais (equivalente às famosas *Máquinas de Turing*—uma função entre números naturais pode ser calculada usando uma Máquina de Turing se, e somente se, ela pode ser escrita no Cálculo Lambda. Logo após as publicações desses resultados, Church veio a ser orientador de doutorado de Turing em Princeton).

No Cálculo Lambda, os objetos sobre os quais tratamos são apenas *funções* (de alta ordem)—todas as outras noções, incluindo os números, são definidos usando funções. Um número natural n, por exemplo, é representado como o conceito abstrato de "dadas uma função f e um argumento x, repetir n vezes a aplicação de f a x". Usando notação de um dos exercícios acima, isso é o mesmo que aplicar f^n a x:

$$\underbrace{f(f(f(\cdots f}_{f \text{ aparece } n \text{ vezes}}\ (x)\cdots))).$$

Assim, cada número natural tem uma contrapartida dentro do Cálculo Lambda, chamada *numeral (de Church)* de n. O numeral correspondente a 0 é a função que podemos escrever em `Python` como:

```
def zero(f):
  def id(x):
    return x
  return id
```

ou, de forma equivalente:

```
zero = lambda f: lambda x: x
```

o numeral correspondente a 1 é:

```
um = lambda f: lambda x: f(x)
```

o numeral correspondente a 2 é:

```
dois = lambda f: lambda x: f(f(x))
```

etc.

a. Faça uma função que receba um natural n e retorne o numeral ao qual ele corresponde.

b. Faça uma função que receba um numeral e retorne o número natural ao qual ele corresponde. Por exemplo, dando a essa função o nome `natural_correspondente` (sinta-se livre para usar o nome que preferir) e usando os exemplos acima, teríamos

```
>>> dois = lambda f: lambda x: f(f(x))
>>> natural_correspondente(dois)
2
```

c. Faça uma função que receba um numeral, correspondente a um natural n, e retorne o numeral correspondente a $n + 1$. Novamente, dando a essa função o nome `sucessor` (sinta-se livre para usar o nome que preferir) e usando os exemplos acima, teríamos

```
>>> dois = lambda f: lambda x: f(f(x))
>>> tres = sucessor(dois)
>>> natural_correspondente(tres)
3
```

d. Faça uma função que receba dois numerais, correspondentes a naturais n e m, e retorne o numeral correspondente a $n + m$.

e. Faça uma função que receba dois numerais, correspondentes a naturais n e m, e retorne o numeral correspondente a $n \cdot m$.

f. Faça uma função que receba dois numerais, correspondentes a naturais n e m, e retorne o numeral correspondente a n^m.