6.2 Funções de ordem superior

1. A função `somatorio`

   ```
   def somatorio(l_inf, l_sup, calc_termo, prox):
       soma = 0
       while l_inf <= l_sup:
           soma = soma + calc_termo(l_inf)
           l_inf = prox(l_inf)
       return soma
   ```

   é apenas a mais simples de um vasto número de abstracções semelhantes que podem ser capturadas por funções de ordem superior. Por exemplo, podemos usar a função `somatorio` para somar os quadrados dos múltiplos de 3 entre 9 e 21:

   ```
   >>> somatorio(9, 21, lambda x : x * x, lambda x : x + 3)
   1215
   ```

   Diga o que fazem as seguintes utilizações da função `somatorio`:

   (a) `somatorio(4, 500, lambda x:  x, lambda x:  x + 1)`

   (b) `somatorio(5, 500, lambda x:  x * x, lambda x:  x + 5)`

   (c)
   ```
   somatorio(1,
             5,
             lambda x: somatorio(1,
                                 x,
                                 lambda x: x,
                                 lambda x: x+1),
             lambda x: x+1)
   ```

2. (a) Defina a função `piatorio` que calcula o produto dos termos de uma função entre dois limites especificados.

   (b) Mostre como definir o `factorial` em termos da utilização da função `piatorio`.

3. Considere a função `soma_fn` que recebe um número inteiro positivo, `n`, e uma função de um argumento inteiro, `fn`, e devolve a soma de todos os valores da função entre `1` e `n`. A função `soma_fn` não verifica a correção do seu argumento nem usa funcionais sobre listas. Por exemplo,

   ```
   >>> soma_fn(4, lambda x: x * x)
   30
   >>> soma_fn(4, lambda x: x + 1)
   14
   ```

   (a) Escreva a função `soma_fn` usando um ciclo `for`.

(b) Escreva a função `soma_fn` usando recursão.

4. Usando recursão, defina os seguintes funcionais sobre listas:

   (a) `filtra(lst, tst)` que devolve a lista obtida a partir da lista `lst` que apenas contém os elementos que satisfazem o predicado de um argumento `tst`. Por exemplo,

   ```
   >>> filtra([1, 2, 3, 4, 5], lambda x : x % 2 == 0)
   [2, 4]
   ```

   (b) `transforma(lst, fn)` que devolve a lista obtida a partir da lista `lst` cujos elementos correspondem à aplicação da função de um argumento `fn` aos elementos de `lst`. Por exemplo,

   ```
   >>> transforma([1, 2, 3, 4], lambda x : x ** 3)
   [1, 8, 27, 64]
   ```

   (c) `acumula(lst, fn)` que devolve o valor obtido da aplicação da função de dois argumentos `fn` a todos os elementos da lista `lst`. Por exemplo,

   ```
   >>> acumula([1, 2, 3, 4], lambda x, y : x + y)
   10
   ```

5. Usando os funcionais sobre listas da pergunta anterior, escreva a função `soma_quadrados_impares`, que recebe uma lista de inteiros e devolve a soma dos quadrados dos seus elementos ímpares. A sua função deve conter apenas uma instrução, a instrução `return`. Não é necessário validar os dados de entrada. Por exemplo:

   ```
   soma_quadrados_impares([1, 2, 3, 4, 5, 6])
   35
   ```

6. Considere a seguinte definição do predicado `eh_primo` que tem o valor verdadeiro apenas se o seu argumento é um número primo:

   ```
   def eh_primo(n):
       if n == 1:
           return False
       else:
           for i in range(2, n):
               if n % i == 0:
                   return False
           return True
   ```

   Escreva a função de ordem superior, `nao_primos` que recebe um número inteiro positivo, `n`, e devolve todos os números inferiores ou iguais a `n` que não são primos. A sua função deve conter apenas uma instrução, a instrução `return`. Por exemplo,

```
nao_primos(10)
[1, 4, 6, 8, 9, 10]
```

7. Considere a seguinte função que recebe como argumentos um número natural, `n`, e um predicado de um argumento, `p`:

```
def misterio(num, p):
    if num == 0:
        return 0
    elif p(num % 10):
        return num % 10 + 10 * misterio(num // 10, p)
    else:
        return misterio(num // 10, p)
```

   (a) Explique o que faz esta função.

   (b) Utilize a função `misterio` para escrever a função `filtra_pares` que recebe um número inteiro e devolve o número obtido a partir dele que apenas contém dígitos pares. A sua função deve conter apenas uma instrução, a instrução `return`. Por exemplo,

```
>>> filtra_pares(5467829)
4682
```

8. Usando funcionais sobre listas, escreva a função `lista_digitos`, que recebe um inteiro positivo `n` e devolve a lista cujos elementos são os dígitos de `n`. A sua função deve conter apenas uma instrução, a instrução `return`. SUGESTÃO: transforme o número numa cadeia de caracteres. Por exemplo:

```
>>> lista_digitos(123)
    [1, 2, 3]
```

9. Usando a função `lista_digitos` do exercício 8 e funcionais sobre listas, escreva a função `produto_digitos`, que recebe um inteiro positivo, `n`, e um predicado de um argumento, `pred`, e devolve o produto dos dígitos de `n` que satisfazem o predicado `pred`. A sua função deve conter apenas uma instrução, a instrução `return`. Por exemplo:

```
>>> produto_digitos(12345, lambda x : x > 3)
    20
```

10. Usando a função `lista_digitos` do exercício 8 e funcionais sobre listas, escreva a função `apenas_digitos_impares`, que recebe um inteiro positivo, `n`, e devolve o inteiro constituído pelos dígitos ímpares de `n`. A sua função deve conter apenas uma instrução, a instrução `return`. Por exemplo: