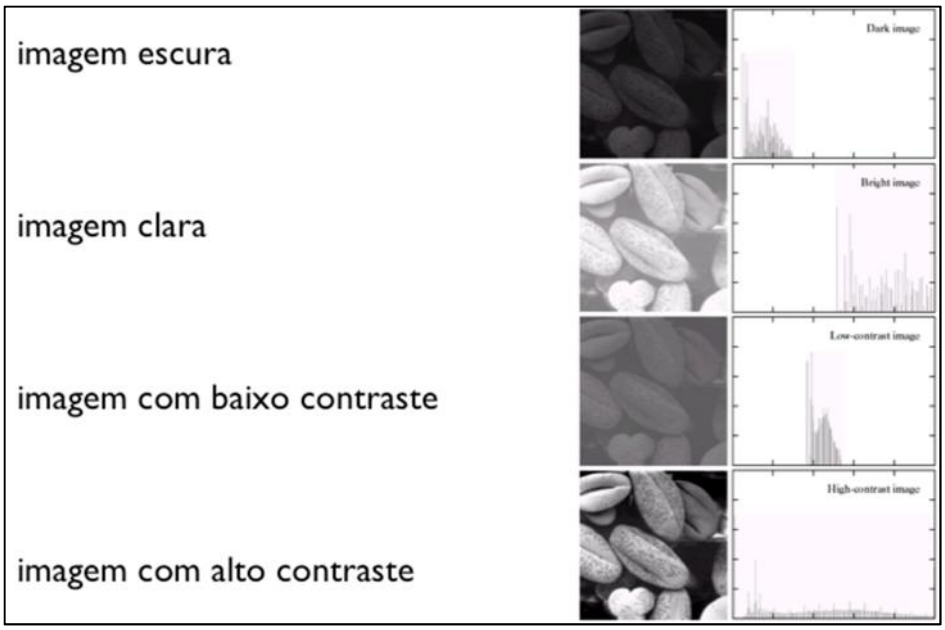
INF100 – Introdução à Programação I
Roteiro Prática 27 a 29 de junho de 2017

Faça o download do arquivo **imagens.pyc** e das imagens (***.jpg**) para dentro da "Pasta pessoal".

Introdução



Em fotografia e processamento digital de imagens, podemos nos deparar com imagens com excesso ou falta de luminosidade, ou então com baixo nível de contraste, como podemos ver nos exemplos ao lado.

Nesta prática, você fará um programa para corrigir uma imagem de baixo contraste. Mas, em vez de lidar com imagens coloridas, faremos isso com uma imagem em tons de cinza. Isso significa que cada *pixel* é formado por um único valor entre 0 (preto) e 255 (branco), em vez dos três componentes R, G, B que usamos na Prática 14. Com isso, podemos usar comandos mais simples para "ler" e "escrever" um *pixel* em uma posição (*i,j*) qualquer:

```
p = im[i][j]
...
im[i][j] = p
```

Roteiro de Prática

Uma visão geral do programa (algoritmo de alto nível) segue abaixo:

1. Leitura e exibição da imagem im1
2. Cálculo do valor mínimo e máximo de pixel presente na imagem
3. Aplicar um Alargamento de Contraste na imagem im1, obtendo a imagem im2
4. Exibição da imagem im2
5. Cálculo e exibição do RMSE entre im1 e im2

O algoritmo acima já está implementado dentro do esqueleto do programa. Para que ele funcione direito, você deverá preencher a implementação das seguintes funções:

`imgMinMax( img )`

Essa função deve percorrer a imagem **`img`** e determinar o menor e o maior valor de *pixel* encontrado nela, retornando esses dois valores. Nos slides da Aula 10 existem exemplos de função que retornam mais de um resultado.

Para percorrer todas as posições (dos *pixels*) de uma imagem **`img`** qualquer, use o modelo já visto nos slides da Aula 8:

```
para i = 0 até img.altura-1:
    para j = 0 até img.largura-1:
        ...
```

alargaContraste(im1, im2, pmin, pmax)

Essa função deve fazer o seguinte:

- Para cada pixel p1 pertencente à imagem **im1**, calcular o pixel p2 correspondente a ser colocado na imagem **im2**, a saber:

```
P2 = round( 255 * (p1 - pmin) / (pmax - pmin))
```

Desdobrando os pixels em suas representações matriciais, temos:

```
im2[i][j] = round( 255 * (im1[i][j] - pmin) / (pmax - pmin))
```

A função `round()` serve para arredondar o valor para o número inteiro mais próximo.

A função deve considerar que a imagem **im2** já foi criada previamente, ou seja, ela só precisa ser preenchida como mostrado acima.

RMSE(f, g)

Essa função deve calcular e retornar uma medida de erro chamada *Root Mean Square Error* (RMSE) entre duas imagens **f** e **g**. Esse cálculo é dado pela seguinte expressão:

$$RMSE = \sqrt{\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(f[i][j]-g[i][j])^2}$$

onde *m* e *n* representam o número de linhas e colunas da imagem, respectivamente.

Obs.: a somatória dupla na equação acima apenas significa que, para cada valor de *i*, devemos percorrer todos os valores de *j*. Para traduzir isso em código Python, usamos um duplo **for**:

Notação matemática	Código em Python

$soma = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A_{ij}$	`soma = 0`
	`for i in range( 0, m ):`
	`  for j in range( 0, n ):`
	`  soma = soma + A[i][j]`

Segue abaixo a saída esperada pelo programa:

```
Calculando min/max...
Alargando contraste...
RMSE entre as imagens 1 e 2: 42.4
```

☞ Não esqueça de preencher o cabeçalho com seus dados e uma breve descrição do programa.

Após certificar-se que seu programa está correto, envie o arquivo do programa fonte (**p17.py**) através do sistema do LBI.