Homework 1

1) Code a method that admits as input a grayscale image **f**, two positive doubles **sx** and **sy** greater than 1, that represent the scaling factors of f and an integer **strat**. The method should return a new image object containing f scaled horizontally by a factor of sx and vertically by a factor of sy.

Example: assume input represents an image of width 100 pixels and height 50 pixels.

```
Image output = scale(input, 2.0, 3.0, strat);
```

should return an image object containing the contents of f, but scaled to a width of 200 and height of 150 pixels. The integer strat represents the interpolation strategy.

Make sure your method supports **nearest neighbor interpolation** (strat = 0) (10 points), **bilinear interpolation** (strat = 1) (20 points), and **bicubic interpolation** (strat = 2) (30 points).

2) Create a method that admits as input a grayscale image **f** and applies histogram equalization to it (10 points).

```
Image output = equalize(input);
```

3) Create another method that performs a local histogram equalization.

```
Image output = equalizeAdaptively(input);
```

First divide the valve.png image regularly into 64 tiles of size 80x60. Then for every tile t, perform a local histogram equalization every tile independently (10 points). You'll notice the output will have border effects between tiles.

4) Now, perform a histogram equalization on every tile t TOGETHER WITH its immediate **3** neighboring tiles (to its east, south and south east). This way the areas you process will overlap, and you'll avoid border effects between tiles. (20 points)

Note 1: you are free to use any software library you like for **loading**, **accessing**, **pixel reading/writing** and **displaying** digital images (e.g. matlab, opencv, etc). Java is recommended.

In case you go with Java, just create a new java project and include the 4 provided jars as external libraries. In this library an image is represented through the class vpt.Image.

Loading an image from your drive:

```
Image img = Load.invoke("path/to/file/myImage.png"); // supports jpg, png
```

Saving an image to your drive:

```
Save.invoke(img, "path/to/save/filename.png");
```

Copy constructor (creates a black image "copy" of the same dimension as "img"):

```
Image copy = img.newInstance(false);
```

Display an Image object "img" visually

```
Display2D.invoke(img, "windowTitle");
```

Reading the pixel intensity of image "img" at **row 300 and column 200**:

```
int p = img.getXYByte(200, 300);
```

Setting the pixel intensity of image “img” at row 300 and column 200 to a new value 128.

```
img.setXYByte(200, 300, 128);
```

Note 2: submit the source code, and the output images that you have obtained for **every** question using the attached valve image in png format.

Good luck.