ESSS Technical Test

The purpose of this test is to see how you would perform in your daily work at ESSS.

Some ground rules:

1. You will be given programming tasks to be done in about 4 hours;
2. The *Suggested Task Duration* (defined on each task) is based on the average time the candidates take to complete the task. It's OK to take more or less time, this is only for your own time management.
3. Allowed programming languages are Python, C++ and Java;
4. Consider this test to be like it was a normal development task:
    a. Feel free to research things in the Internet, like you would do you in your daily work.
    b. Feel free to ask us if you have any questions, just like you would ask to a colleague in your daily work.
    c. Document the code and add comments as you see fit.
5. After finishing each task send the source code to tech-eval@esss.com.br and tell us via Skype so we can take a look and chat with you about the code.
6. In the examples below we use Python in the command line, but adapt it to Java or C++ as needed.

Task 1 - RGB Image Decomposition

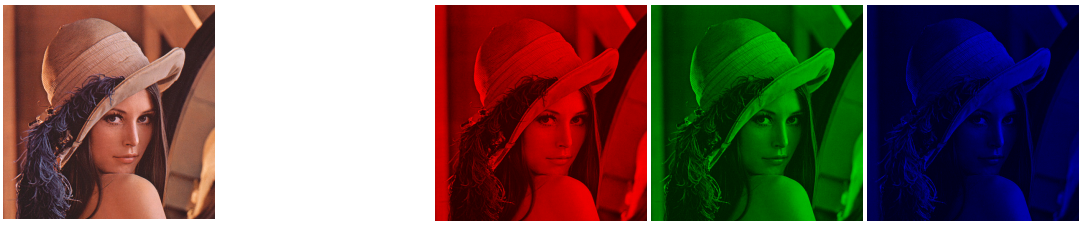
Suggested Task Duration: 1h

Given an **PNG** image file, write a program that creates 3 new image files, each one having a single RGB color channel from the original image.

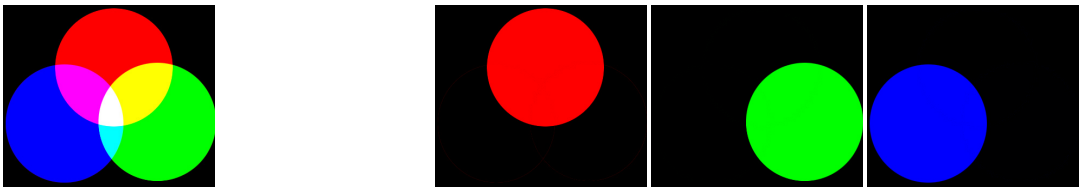
RGB Decomposition algorithm

An image file can be seen as a NxM pixel matrix (where N is the image width and M is the height). Each pixel is represented by 3 values (R, G, B) where each value defines the color code of each color channel. The algorithm must create 3 new images, each image must have the original value for a single RGB channel and 0 (zero) for the other channels. The resulting images should be as follows:

Example 1:



Example 2:



Program should be executable via command line, as follow:

```
$ python esss_test.py <image-file>
```

Notes:

- You should use *any external library* of your choice to read/write the image files, although the decomposition algorithm **must** be implemented by yourself; this is valid for the next tasks as well;
- Getting the library to “work” should take no more than 15 minutes. *Ask for help* if it’s taking more than that!
- *If C++ is chosen*, take care to not spend too much time getting an image library to work, prefer installing binaries if possible. *If on C++ on Windows*, we recommend `OpenCV` installed from binaries together with MSVC 2015, or `Qt` used from Qt Create, instead of building from source yourself.
- Libraries suggestion for Python: `Pillow` or `scikit-image`.

- In case you don’t know where to start, take a look at the code snippets below

Python

```python
import sys
from PIL import Image

img = Image.open(sys.argv[1])
width, height = img.size
for x in range(width):
    for y in range(height):
        r, g, b, alpha = img.getpixel((x, y))
        print("R: {:3}, G: {:3}, B: {:3}".format(r, g, b))
```

Java

```java
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class Main {

    public static void main(String[] args) {
        BufferedImage image = null;
        try {
           image = ImageIO.read(new File(args[0]));
        } catch (IOException e) {
            System.err.println(e.getMessage());
            return;
        }

        int w = image.getWidth();
        int h = image.getHeight();
        for(int y = 0; y < image.getHeight(); ++y) {
            for (int x = 0; x < image.getWidth(); ++x) {
                Color pix = new Color(image.getRGB(x, y));;
                System.out.println(
                    " R: " + pix.getRed() +
                    " G: " + pix.getGreen() +
                    " B: " + pix.getBlue()
                    );
            }
        }
    }
}
```

C++

```cpp
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/opencv.hpp>

int main(int argc, char* argv[])
{
  if (argc != 2) {
    std::cout << "Wrong number of arguments." << std::endl;
    return 1;
  }

  char *filename = argv[1];
  cv::Mat image = cv::imread(filename, cv::IMREAD_COLOR);

  if (!image.data) {
    std::cout << "Could not open the image file." << std::endl;
    return 2;
  }

  for (int i = 0; i < image.rows; i++) {
    for (int j = 0; j < image.cols; j++) {
      cv::Vec3b pixel = image.at<cv::Vec3b>(i, j);
      std::cout << "R: " << pixel[2] << ", G: " << pixel[1] << ", B: " << pixel[0] << std::endl;
    }
  }

  return 0;
}
```