# Homework 1 Questions

## Instructions

- Compile and read through the included MATLAB tutorial.
- 2 questions.
- Include code.
- Feel free to include images or equations.
- Please make this document anonymous.
- **Please use only the space provided and keep the page breaks.** Please do not make new pages, nor remove pages. The document is a template to help grading.
- If you really need extra space, please use new pages at the end of the document and refer us to it in your answers.

## Submission

- Please zip your folder with **hw1_student id_name.zip** (ex: hw1_20181234_Peter.zip)
- Submit your homework to KLMS.
- An assignment after its original due date will be degraded from the marked credit per day: e.g., A will be downgraded to B for one-day delayed submission.

# Questions

**Q1:** We wish to set all pixels that have a brightness of 10 or less to 0, to remove sensor noise. However, our code is slow when run on a database with 1000 grayscale images.

*Image:* grizzlypeakg.png

```
A = imread('grizzlypeakg.png');
[m1,n1] = size( A );
for i=1:m1
    for j=1:n1
        if A(i,j) <= 10
            A(i,j) = 0;
        end
    end
end
```

**Q1.1:** How could we speed it up?

**A1.1:** Your answer here.

We could use Logical Indexing method here.

```
A = imread('grizzlypeakg.png');
B = A <= 10;
A(B) = 0;
```

**Q1.2:** What factor speedup would we receive over 1000 images? Please measure it.

Ignore file loading; assume all images are equal resolution; don't assume that the time taken for one image $\times 1000$ will equal 1000 image computations, as single short tasks on multitasking computers often take variable time.

**A1.2:** Your answer here.

tic; toc; method is useful for calculating the time spended with both type of code.

The case of naive algorithm:

```
A = imread('grizzlypeakg.png');
tic;
for k=1:1000
    [m1,n1] = size( A );
    for i=1:m1
        for j=1:n1
            if A(i,j) <= 10
                A(i,j) = 0;
            end
        end
    end
end
toc;
```

The case of effective algorithm:

```
A = imread('grizzlypeakg.png');
tic;
for k=1:1000
    B = A <= 10;
    A(B) = 0;
end
toc;
```

Naive algorithm took 24.91 seconds on average, but the effective algorithm took 9.62 seconds on average. (Macbook Retina 13inch, Early 2015) So the effective algorithm is likely 2.59 times faster than the naive algorithm.

**Q1.3:** How might a speeded-up version change for color images? Please measure it.

*Image:* grizzlypeak.jpg

**A1.3:** Your answer here.

Logical Indexing could be used in 3D array too. We can seperate the image into R, G, B matrices, then merge them and use the result to remove the noise.

```
A = imread('grizzlypeak.jpg');
R = A(:,:,1) <= 10;
G = A(:,:,2) <= 10;
B = A(:,:,3) <= 10;
RGB = cat(3, R, G, B);
A(RGB) = 0;
```

**Q2:** We wish to reduce the brightness of an image but, when trying to visualize the result, all we sees is white with some weird "corruption" of color patches.

*Image:* gigi.jpg

```
I = double( imread('gigi.jpg') );
I = I - 20;
imshow( I );
```

**Q2.1:** What is incorrect with this approach? How can it be fixed while maintaining the same amount of brightness reduction?

**A2.1:** Your answer here.

It is not correct using `double( imread('gigi.jpg') )` to convert image to double type. We can use `im2double` function to convert correctly.

```
I = imread('gigi.jpg');
I = im2double(I);
I = I - 20/255;
imshow(I);
```

**Q2.2:** Where did the original corruption come from? Which specific values in the original image did it represent?

**A2.2:** Your answer here.

This is the problem of type casting.
`im2double` function converts 0-255 range to 0-1 range,
but `double( imread('gigi.jpg') )` do not.
So if `value - 20` is 1 or over, it would be displayed as white. If not, we can see other colors.
For example, if we have a pixel with `(21, 19, 15)` as RGB values,
they will be converted to `(1, -1, -5)` which means true red color.