Digital Image Processing
Application Training Exercise - AY1819
UP Digital Signal Processing Laboratory

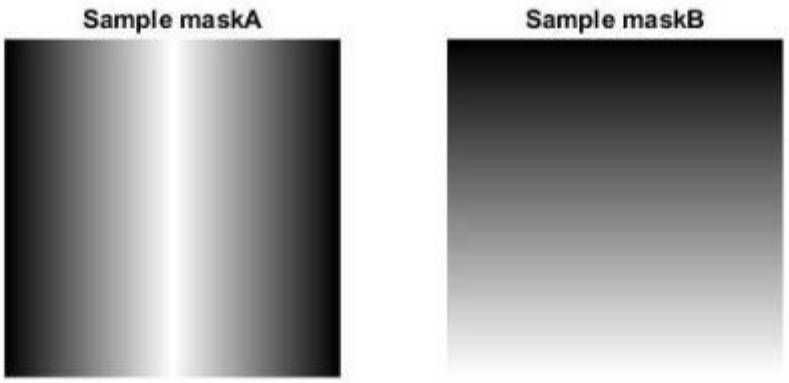
General Instructions:

1. This training exercise should be done **individually**.
2. You are required to submit a **bug-free** MATLAB/Octave script containing answers for all the tasks/problems. Write your **name**, your **student number**, the **date** of submission and the MATLAB/Octave **version** that was used as comments in every script you create. Separate each subtask into sections in your script. Create a script for every task with the filename following the format **<Task#>_Surname.m**
3. Provide explanations or details in your code as comments.
4. Put appropriate titles and labels on your displayed images and graphs.
5. The deadline for submission is on **May 5, 2019, 23:59**.
6. Compress all your files (scripts, images) into a .zip file named **DIP_Exercise_<Surname>.zip**
7. Send your submissions to **paula.yap@eee.upd.edu.ph** with the subject **DIP Exercise <Surname>**
8. Good luck, and have fun!

Task #1: Horizontal and Vertical Image Blending (Introduction to Masking)

Masking is a useful tool in image processing and will be necessary to do both spatial and frequency domain enhancements. For this task, **choose any two input images of the same size.**

1. First, create two masks called *maskA* and *maskB*. The masks should look like the images below and have a uniform gradient from white to black. **The masks should have the same dimensions as the input images.**



   Hint: `repmat()` can be used copy the contents of a single row across multiple columns

2. Use element-wise multiplication to apply *maskA* to the first input image and *maskB* to the second input image. Name the results *maskedA* and *maskedB* respectively.

3. Blend *maskedA* and *maskedB* using a linear combination of the gray level values as in the following:

$$h(x, y) = \alpha \cdot f_A(x, y) = (1 - \alpha) \cdot f_B(x, y)$$

   Use an alpha value of 0.5. Name the resulting image as *blended*.

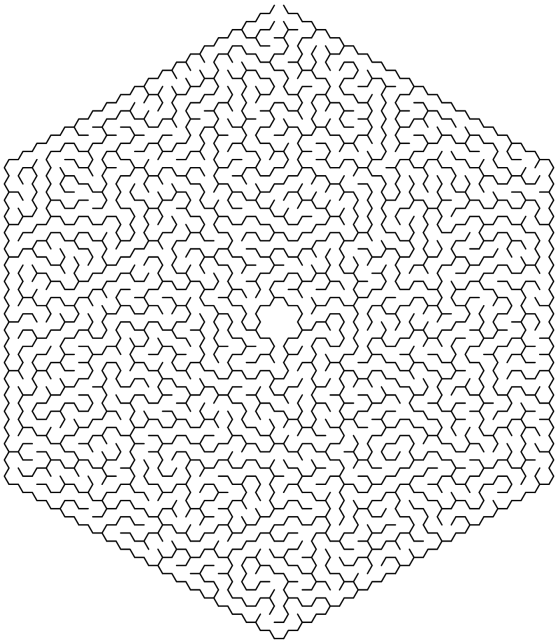
4. Display *maskedA*, *maskedB*, and *blended* using `subplot()`.

5. Save *maskedA*, *maskedB*, and *blended* as .png files with the same name.

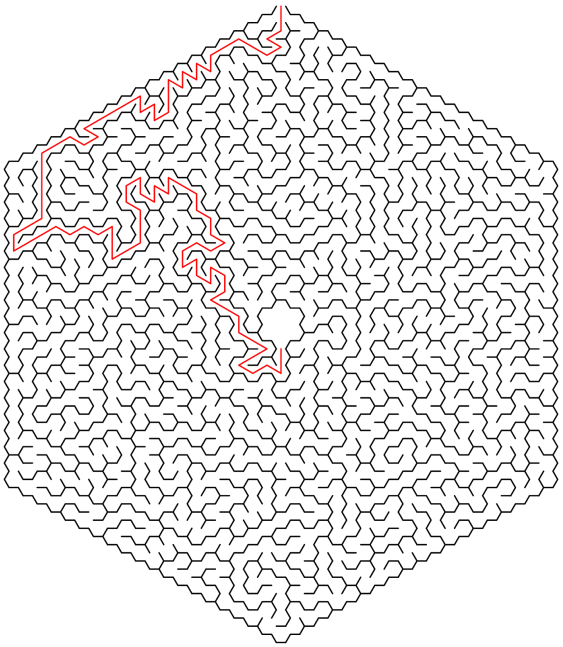
Task #2: Maze Solver

Design and implement an algorithm to solve the attached hexagonal maze (**maze.png**). The algorithm should be able to connect the white region on the **outside** of the hexagon with the white space in the **middle** of the hexagon. The output of the algorithm should be similar to the attached solution (**soln.png**). Save the output of the algorithm as a .png file. Include a detailed explanation of your algorithm as comments.

Hint: `bwmorph()` *is a multi-purpose function used for morphological operations on black-and-white images.*



maze.png



soln.png