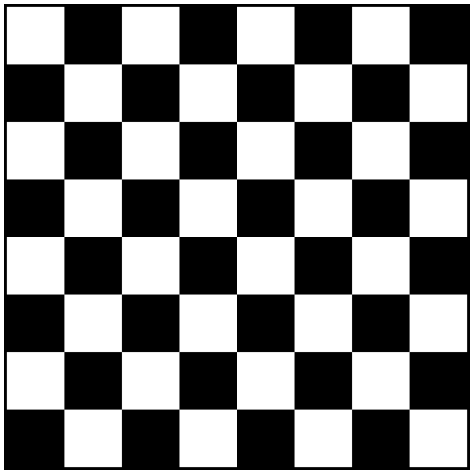
Checkerboard

[**Question 1.**] Starting from a blank image that is 800 by 800 pixels, create the traditional black and white checkerboard! Note there are a total of 64 squares (8 by 8) where the first square on each row alternates between black and white.



Program Outputs

- *Use **imshow or imwrite** to check your checkerboard!*

Image Puzzles

[**Question 2.**] Given a jumbled image, correct the pixel data to reveal the true image. (Special thanks to Dr. Parlante from Stanford for the original idea!)

Key programming concepts: image processing, 3D matrices, imread, imwrite
Approximate lines of code: 24 (does not include comments or white space)

Program Inputs

- **Enter image name:**
  - *The user will always enter a png image with the phrase* ***_Puzzle.png*** *at the end*

Program Outputs

- **Corrected image saved to XXX**
  - *Replace the word* ***Puzzle*** *with* ***Fixed*** *in the original name, saving the new image to file*

Assignment Details:

A digital image is essentially a multidimensional collection of pixels, which are the smallest components of the image. Think of an image as a closely-packed grid of dots that each have their own color, represented using the RGB format: Red-Green-Blue. By combining thousands or millions of these pixels, we can create rich digital images! Your goal for this assignment is to solve different image puzzles by implementing a **pixel decoding algorithm**. For each puzzle, a real image was taken and the pixels scrambled to hide the true image. Check out the following example for the Statue of Liberty:

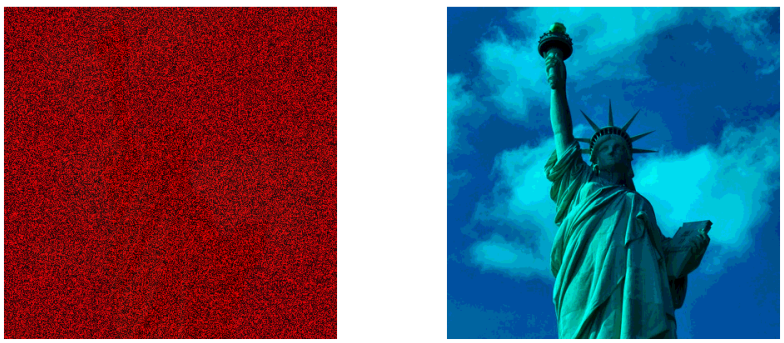


Figure 1: Statue of Liberty image scrambled by lowering the blue and green pixel values.

Sample Output

After applying the pixel decoding algorithm, you should get a recognizable image back (Note that the colors of the image will no longer be perfect due to the algorithm).

Test Case 1:

```
Enter image name: Cool_Puzzle.png
Corrected image saved to Cool_Fixed.png
```

Test Case 2:

```
Enter image name: Excited_Puzzle.png
Corrected image saved to Excited_Fixed.png
```

Test Case 3:

```
Enter image name: Fun_Puzzle.png
Corrected image saved to Fun_Fixed.png
```

Pixel Decoding Algorithm

Use the following steps to solve the puzzle:

- Create a blank image of the same size as the image puzzle
- Go to every row and column location in the image and do the following:
  - Move the current pixel to a new row and column location in the blank image:
    - If at an **even row index**, use the formula: $new\ row\ =\ row/2$
    - If at an **odd row index**, use the formula: $new\ row\ =\ (row\ +\ 1)/2\ +\ (image\ height)/2$
    - If at an **a col in the left half**, use the formula: $new\ col\ =\ (image\ width)/2 + 1 - col$
    - If at an **a col in the right half**, use the formula: $new\ col = 3*(image\ width)/2 + 1 - col$
  - Fix the colors in the image:
    - Multiply all the red and blue pixel values by 3
    - Set all the green pixel values to 0
- Rearrange the image:
  - Split the image in half horizontally and vertically to create 4 quadrants
  - Fix the quadrants by rotating each to the next quadrant in the clockwise direction
- Save the fixed image to a file where the word **Puzzle** has been replaced by **Fixed**