# Blending

## Introduction

In this assignment, we will be putting together a pyramid blending pipeline that will allow us to turn the following three images into a blended image

The blending procedure takes two images and the mask and splits them into their RGB channels.
The code will construct a laplacian pyramid for the two images and scale the mask image to the range [0,1] and construct a Gaussian pyramid for it.

## Reduce and Expand functions.

### reduce

This function takes an image, convolves it, and then subsamples it down to a quarter of the size (dividing the height and width by two). Note that we say subsample here. We recommend you look into numpy indexing techniques to accomplish the subsampling (essentially you want to index every other row and every other column).
Within the code, you are provided with a generating_kernel(a) function. This function takes a floating point number a, and returns a 5x5 generating kernel. For the reduce and expand functions, you should use **a = 0.4.**
Seeing as we have already covered how convolve works in class, for this assignment we allow you to use the scipy library implementation of convolve.

```
scipy.signal.convolve2d(image, kernel, 'same')
```

This call will convolve the image and kernel, and return an array of the 'same' size as image.

### expand

This function takes an image and supersamples it to four times the size (multiplying the height and width by two (2)). After increasing the size, we have to interpolate the missing values by using the same convolution we used in reduce, and lastly we scale our output by 4.

For this part of the assignment, please use the generating kernel with **a = 0.4** and the convolve2d function from the reduce function discussion above.

## Gaussian and Laplacian Pyramids

In this part of the assignment, you will be implementing functions that create Gaussian and Laplacian pyramids
In addition, assignment4_test.py defines the functions viz_gauss_pyramid and viz_lapl_pyramid, which take a pyramid as input and return an image visualizaiton. You might want to use these functions to visualize your pyramids while debugging -- we use these at the end of the testing to output your results!

### gaussPyramid

This function takes an image and builds a pyramid out of it. The first layer of this pyramid is the original image, and each subsequent layer of the pyramid is the reduced form of the previous layer. Put simply, you are iteratively calling the reduce function on the output of the previous call, with the first call simply being the input image.
Please use the reduce function that you implemented in the previous part in order to write this function.

### laplPyramid

This function takes a Gaussian pyramid constructed by the previous function, and turns it into a Laplacian pyramid. The doc string contains further information about the operations you should perform for each layer.
Like with Gaussian pyramids, Laplacian pyramids are represented as lists of numpy arrays in the code.
Please use the expand function that you implemented in the previous part of the code in order to write this function.

## Writing the blend and collapse functions.

In this part, you will be completing the pipeline by writing the actual blend function, and creating a collapse function that will allow us to convert our Laplacian pyramid into an output image.

### blend

This function takes three pyramids:

- white - a laplacian pyramid of an image
- black - a laplacian pyramid of another image.
- mask - a gaussian pyramid of a mask image.

It should perform an alpha-blend of the two Laplacian pyramids according to the mask pyramid. So you will be blending each pair of layers together using the mask of that layer as the weight. As described in the doc string, pixels where the mask is 1 should be taken from the white image, pixels where the mask is 0 should be taken from the black image. Pixels with value 0.5 in the mask should be an equal blend of the white and black images. This is mathematically described in the assignment comments.
You may assume that all of the provided pyramids are of the same dimensons, and have dtype float. You may further assume that the mask pyramid has values in the range [0,1].

Your output pyramid should be of the same dimensions as all the inputs, and dtype float.

### collapse

This function is given a laplacian pyramid and is expected to 'flatten' it to an image.
We need to take the bottom (smallest) layer, expand it, and then add it to the next layer. We continue this process until we reach the top of the pyramid. Once you add the second layer to the top layer, the top layer is your output (a common mistake is to re-add the top layer to the top layer so if your values look like they are twice as big as what the expected output is, that is what you are doing).
This function should return a numpy array of the same shape as the top (largest) layer of the pyramid, and dtype float.

## What To DO?

Bonus:
Choose a black image and a white image that you would like to blend.
Choose a unique mask (don't use the one we provide) that you created and explain how you created it.