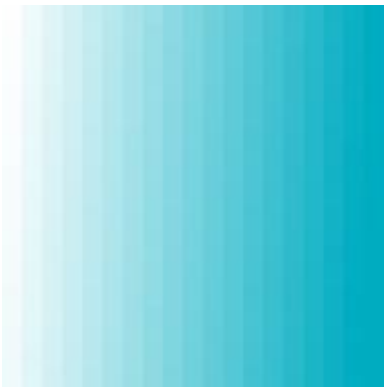
**Image Processing**

Part 1: Image Processing methods-

We created a (1) RGBImage class and a (2) ImageProcessing class. The RGBImage class enabled us to initialize new instances of images in RGB color spaces. The ImageProcessing class enabled us to execute changes to the RGBImage instances.

These are the original pictures that we had:



After executing the rotate 180 degrees method, this image is created:

After executing the crop method, this image is created:

After executing the negate method, this image is created:


After executing the chroma key (overlap) method with the original two images, this image is created:


After executing the tint method, this image is created:

After executing the clear channel method (clearing two different intensity channels), these images are created:

Part 2: Modified KNN classifier-

We created a simple KNN classifier (ImageKNNClassifier class) that takes in training data (pictures with their labels) and a test image to predict the label of the test image.

We created random images with “low” and “high” intensity values as the training data. Then, we created low and high intensity images and tested to see if our KNN classifier worked as intended:

```
def knn_test_examples():
    """
    Examples of KNN classifier tests.
    """
    # make random training data (type: List[Tuple[RGBImage, str]])
    train = []
    # create training images with low intensity values
    train.extend(
        (RGBImage(create_random_pixels(0, 75, 300, 300)), "low")
        for _ in range(20)
    )
    # create training images with high intensity values
    train.extend(
        (RGBImage(create_random_pixels(180, 255, 300, 300)), "high")
        for _ in range(20)
    )

    # initialize and fit the classifier
    knn = ImageKNNClassifier(5)
    knn.fit(train)

    # should be "low"
    print(knn.predict(RGBImage(create_random_pixels(0, 75, 300, 300))))
    # can be either "low" or "high"
    print(knn.predict(RGBImage(create_random_pixels(75, 180, 300, 300))))
    # should be "high"
    print(knn.predict(RGBImage(create_random_pixels(180, 255, 300, 300))))
```