**Lab 7:** Cassandra Cabrera and Mike Menendez **Due:** Feb.17, 2020

**Summary:** The hardest part of this was task 4. The change on the threshold to detect the color distance required a lot of extra work and was not very compute efficient. In order to optimize it, we attempted to convert the images to c arrays via numpy and to vectorize an abstract function that would allow us to create a merged vector off of a predetermined threshold. Unfortunately due to the limitations of working with two individual arrays, it became difficult to iterate them properly. This resulted in minor tweaks to the n^2 solution to optimize it where possible. Once the solution was deemed fast enough, we then utilized a sudo grid search which consisted of using a print statement to find the color distances for each pixel from the green value. Once we had that, we then redirected the print output to a temp file which was then piped to the gnu sort and unique with the -c flag which is for counting the unique values. This allowed us to find a manual color threshold which allowed us to get the final result with an extremely minimal green outline. If we were allocated additional time, we could potentially get a better result that would only be able to be more precise with manual editing of the images.

## Task1:

```python
from PIL import Image

"""
Task 1:
    - Return a coordinate of a pixel with the highest red value (can be multiple, just return one)
    - Steps:
        - Read in image
        - Traverse Image as if it were a 3D matrix (x, y, 0 (where 0 = red channel in rgb tuple))
        - Create hashmap to store values based off red value (dynamic programming bruh :fire:)
        - Get keys from hashmap, reverse sort and display the list of the max value
"""


def main():
    print("Enter file path:")
    img = Image.open(input().strip(), 'r')
    mappy = {}
    maxxR = 0
    coor = (0,0)
    for x in range(img.width):
        for y in range(img.height):
            if img.getpixel((x,y))[0] > maxxR:
                maxxR = x
                coor = (x,y)
    print("Coordinate of pixel with the max red channel: ", coor)


if __name__ == "__main__":
    main()
```

## Results from Task1:

```
(env) Cassandras-MacBook-Pro:7 casscabrera$ python3 task1.py
Enter file path:
/Users/casscabrera/Desktop/shiny-potato/labs/7/reddest.jpg
Coordinate of pixel with the max red channel:  (286, 2339)
(env) Cassandras-MacBook-Pro:7 casscabrera$
```

## Task2:

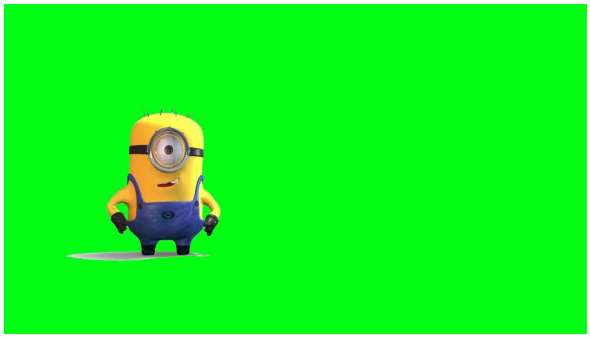
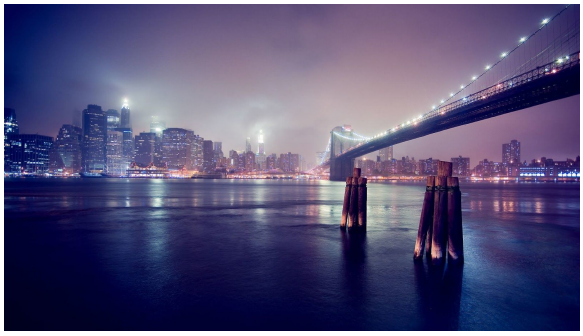
```python
from PIL import Image

"""
Task 2:
    - Pick three (small) images and place them on a blank canvas such that they do not overlap.
    - Steps:
        - Read in images
        - Traverse images one by one, plotting their pixels.
        - Print Newly created image
"""

def main():
    print("Enter first image path:")
    img1 = Image.open(input().strip(), 'r')
    print("Enter second image path:")
    img2 = Image.open(input().strip(), 'r')
    print("Enter third image path:")
    img3 = Image.open(input().strip(), 'r')

    canvas_x = img1.width + img2.width + img3.width
    canvas = Image.new("RGB", (canvas_x,img1.height), "white")

    target_x = 0
    for source_x in range(img1.width):
        target_y = 0
        for source_y in range(img1.height):
            color = img1.getpixel((source_x, source_y))
            canvas.putpixel((target_x, target_y), color)
            target_y += 1
        target_x +=1

    for source_x in range(img2.width):
        target_y = 0
        for source_y in range(img2.height):
            color = img1.getpixel((source_x, source_y))
            canvas.putpixel((target_x, target_y), color)
            target_y += 1
        target_x +=1

    for source_x in range(img3.width):
        target_y = 0
        for source_y in range(img3.height):
            color = img1.getpixel((source_x, source_y))
            canvas.putpixel((target_x, target_y), color)
            target_y += 1
        target_x +=1

    canvas.show()



if __name__ == "__main__":
    main()
```

## Results from Task2:

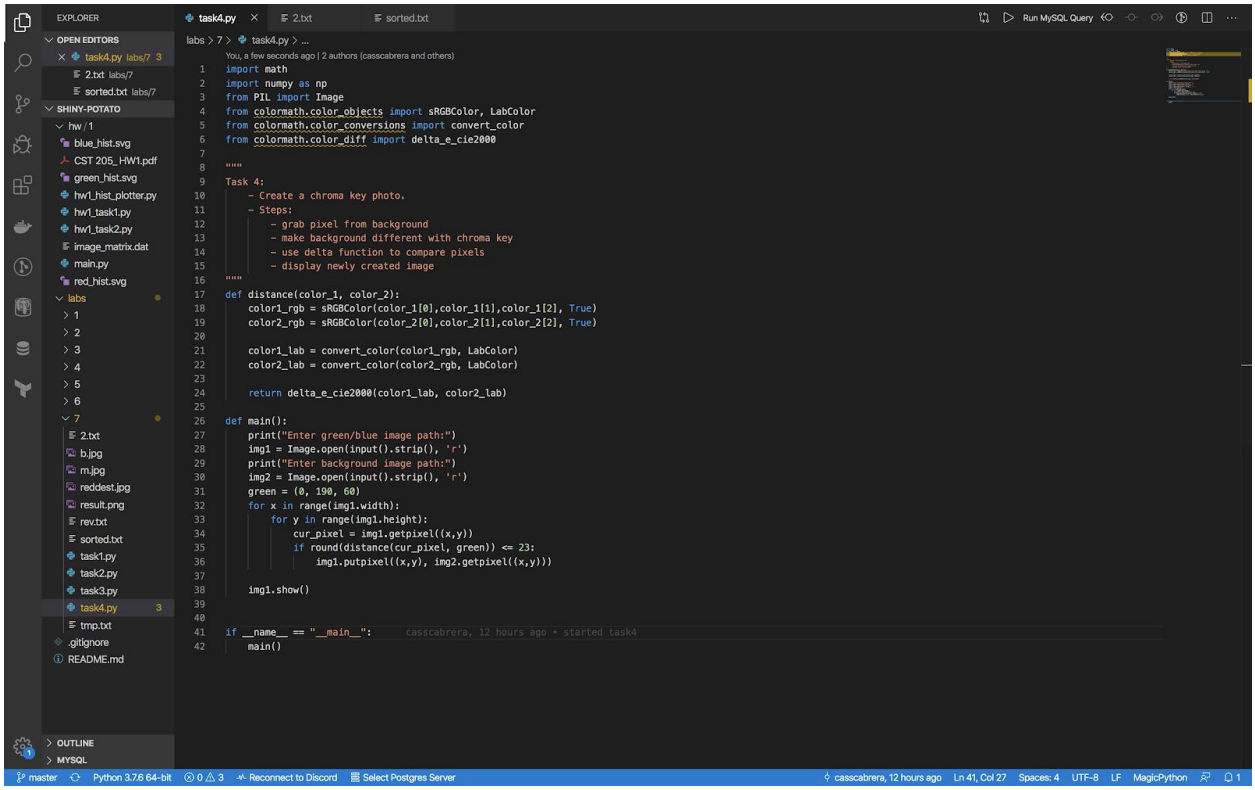
## Task3:

```python
import math
from PIL import Image

"""
Task 3:
    - Create a chroma key photo.
    - Steps:
        - grab pixel from background
        - make background different with chroma key
        - display newly created image
"""
def distance(color_1, color_2):
    red_diff = math.pow((color_1[0] - color_2[0]), 2)
    green_diff = math.pow((color_1[1] - color_2[1]), 2)
    blue_diff = math.pow((color_1[2] - color_2[2]), 2)
    return math.sqrt(red_diff + green_diff + blue_diff)

def main():
    print("Enter green/blue image path:")
    img1 = Image.open(input().strip(), 'r')
    print("Enter background image path:")
    img2 = Image.open(input().strip(), 'r')

    for x in range(img1.width):
        for y in range(img1.height):
            cur_pixel = img1.getpixel((x,y))
            green = (0, 190, 60)
            if distance(cur_pixel, green) < 150:
                img1.putpixel((x,y), img2.getpixel((x,y)))

    img1.show()


if __name__ == "__main__":
    main()
```

## Results from Task3:


+

=

# Task4:



```python
import math
import numpy as np
from PIL import Image
from colormath.color_objects import sRGBColor, LabColor
from colormath.color_conversions import convert_color
from colormath.color_diff import delta_e_cie2000

"""
Task 4:
    - Create a chroma key photo.
    - Steps:
        - grab pixel from background
        - make background different with chroma key
        - use delta function to compare pixels
        - display newly created image
"""
def distance(color_1, color_2):
    color1_rgb = sRGBColor(color_1[0],color_1[1],color_1[2], True)
    color2_rgb = sRGBColor(color_2[0],color_2[1],color_2[2], True)

    color1_lab = convert_color(color1_rgb, LabColor)
    color2_lab = convert_color(color2_rgb, LabColor)

    return delta_e_cie2000(color1_lab, color2_lab)

def main():
    print("Enter green/blue image path:")
    img1 = Image.open(input().strip(), 'r')
    print("Enter background image path:")
    img2 = Image.open(input().strip(), 'r')
    green = (0, 190, 60)
    for x in range(img1.width):
        for y in range(img1.height):
            cur_pixel = img1.getpixel((x,y))
            if round(distance(cur_pixel, green)) <= 23:
                img1.putpixel((x,y), img2.getpixel((x,y)))

    img1.show()


if __name__ == "__main__":
    main()
```

# Results from Task4: