UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA - UTEC

Tarea 3

Fecha de inicio: *25 de Mayo, 2022*
Fecha de entrega: *12 de Junio, 2022*

Curso: *Programación 1 (1100)* – Entrega mediante: *Gradescope*

Indicaciones generales

1. Recuerda que la tarea es **individual**. Los casos de copia/plagio serán sancionados con nota cero (0) en la asignatura.

2. (a) La fecha límite de entrega es el **domingo 12 de Junio a las 23:59 hrs**.
   (b) Es altamente recomendable no esperar hasta la última hora.
   (c) *Gradescope* desactivará automáticamente los envíos pasada dicha hora límite.
   (d) **No se aceptarán entregas atrasadas ni entregadas por otros medios.**

3. Revisa bien lo que entregas, aunque en esta oportunidad podrás entregar ilimitadas veces la tarea, la última enviada será la evaluada.

4. Recuerda que *Gradescope* corrige automáticamente tu entrega. Dicha plataforma mostrará si has realizado correctamente las pruebas y mostrará algunos mensajes en color verde. Puedes ver un ejemplo de este caso en el anexo 6.

5. Es posible que hayas subido tu entrega pero hayas modificado algo que no se debió en el template. En ese escenario, Gradescope te mostrará algunos mensajes de error. Puedes ver un ejemplo de esto en el anexo 7.

Gradescope

1. Nosotros les proporcionaremos un código base de donde deberán partir para completar dicho ejercicio. Este archivo es llamado `solution.py` y lo encontrarán en la indicación de la tarea en CANVAS.

2. Al finalizar, **solo** subir el archivo `solution.py` (NO cambiar el nombre del archivo y NO comprimirlo).

3. Cada pregunta tiene diversos casos de prueba. Para obtener la nota completa en una pregunta, el algoritmo debe obtener la respuesta correcta en dichos casos de prueba.

4. Si un caso de prueba falla, visualizarán un mensaje de error con sugerencias. **Lee el error**, revisa el código e inténtalo de nuevo.

5. Los input de los casos de prueba son confidenciales.

Indicaciones específicas

1. En el anexo B, se puede ver la plantilla de código.

2. Ustedes deben escribir dentro de la sección y a la misma altura de donde esta escrito *"Código comienza aquí"*. Además, no deben modificar nada debajo de *"Código acaba aquí"*. Recuerden tener cuidado con las indentaciones.

3. Los input del ejercicio se encuentran en la plantilla. Recuerden usar estas variables para resolver el ejercicio.

4. Ustedes podrán utilizar la imagen 'programacion.bmp' para hacer la prueba de su implementación. Es importante que esta imagen se encuentre en la misma dirección que su solution.py. Si desean utilizar otra imagen, tiene que tener un formato BMP de 24 bits.

5. La respuesta de los ejercicios debe ser retornada en una lista de tres dimensiones, el mismo formato de imágenes vistas en clase de matrices gráficas, según se encuentra especificado en la plantilla otorgada.

6. Para resolver esta tarea van a tener que utilizar las funciones `leer_imagen` y `guardar_imagen` con el objetivo de convertir la imagen en formato BMP a una lista tridimensional y de una lista tridimensional a una imagen en formato BMP.

7. Si realizan cálculos donde el resultado es un número con decimales luego de aplicar una operación matemática, tienen que utilizar la función `round` para redondearlo al entero más cercano o asegurarse que su operación retorne un valor de tipo *int* usando la operación de división entera antes de guardarlo en la lista final.

Para la resolución de esta tarea, se utilizará la siguiente imagen como referencia:

Figure 1: Representación de la imagen original

Se solicita que realicen cambios a la imagen original para que se vea de distintas formas. Por lo cual en cada pregunta se implementarán diferentes filtros que modifiquen la imagen original.

Pregunta 1: Filtro Negativo - (5 pts)

En este filtro se quiere convertir la imagen original de colores a un modo negativo. Para que la imagen se convierta en negativo se tiene que restar el color blanco en su formato RGB (255,255,255), con el color de cada píxel de la imagen original. Por lo que, si todo los pixeles cumplen esta condición, la imagen se verá negativa.

Por ejemplo, dada una matriz de tres dimensiones:

$$\begin{pmatrix} [83,118,194] & [31,130,58] \\ [222,194,38] & [252,10,252] \end{pmatrix}$$

al sustraer los colores originales al color blanco:

$$\begin{pmatrix} [255,255,255]-[83,118,194] & [255,255,255]-[31,130,58] \\ [255,255,255]-[222,194,38] & [255,255,255]-[252,10,252] \end{pmatrix}$$

Se generará una nueva matriz tridimensional con filtro negativo.

$$\begin{pmatrix} [172,137,61] & [224,125,197] \\ [33,61,217] & [3,245,3] \end{pmatrix}$$

Después de aplicar el filtro tendremos como resultado la siguiente imagen:

Figure 2: Representación de la imagen negativa

Pregunta 2: Filtro Forma Circular - (5 pts)

Para este ejercicio se va a hacer un filtro de forma circular. Para ello, se necesita que el círculo se encuentre en el centro de la imagen con coordenadas Cx, Cy. Para hacer la forma circular deben de utilizar la fórmula

$$(j - C_x)^2 + (i - C_y)^2 < \left(\frac{h}{2}\right)^2 \tag{1}$$

donde h es la altura de la imagen, w es el ancho de la imagen, i y j son la i-ésima fila, dado $0 \leq i \leq h - 1$, y j-ésima columna, dado $0 \leq j \leq w - 1$, de la matriz. Esta fórmula da como resultado un pixel que cae dentro del círculo.

Observaciones. Todo lo que esté dentro del circulo debe ser la imagen original, lo demás debe de ser el color negro.

Por ejemplo, dada una imagen de 720×720, su centro sería

$$C_x = 360, C_y = 360.$$

Para calcular si un pixel cualquiera cae dentro del círculo, debemos ver su posición, $i = 200, j = 120$, y al reemplazar en la fórmula:

$$(120 - 360)^2 + (200 - 360)^2 < \left(\frac{720}{2}\right)^2$$

, obtenemos como resultado

$$83200 < 129600$$

Al ser cierta la afirmación podemos indicar que el pixel ubicado en $i = 200, j = 120$ cae dentro del círculo, por lo que el pixel debe tener su color original. Si hubiese sido falsa la afirmación debería de tener el color negro en formato RGB(0,0,0).

Figure 3: Representación de la imagen con corte circular

Pregunta 3: Filtro Reflejo Vertical con formato BGR - (5 pts)

El objetivo de aplicar este filtro es rotar la imagen verticalmente por lo que cualquier pixel que se encuentre en la posición de abajo debería de ir arriba y los que están en la posición de arriba deberían de ir abajo. Además, a este filtro también se le tiene que cambiar del formato RGB (rojo, verde, azul) al formato BGR (azul, verde, rojo). Es decir, los colores que se encuentran en la posición R pasan a la posición B y los colores que se encuentran en la posición B pasan a la posición R. Solo los colores que se encuentran en la posición G mantienen su posición.

Por ejemplo:

$$[235, 45, 67] \rightarrow [67, 45, 235] \tag{2}$$

En este ejemplo, se tiene el valor de un pixel de la imagen original en el formato RGB. Por ello el valor númerico de la escala en rojo es 235, el valor del verde es 45 y el valor del azul es 67. Entonces, para cambiarlo al formato BGR, se tiene que cambiar el valor de rojo por el valor del azul y el valor del azul por el valor del rojo. Lo que resultaría en que la imagen final tendría como valor númerico de la escala en rojo a 67 y la escala en azul ahora su valor numérico sería 235. El único color que mantiene su

valor númerico es el verde. Lo que esto convertiría a la imagen final en un formato BGR.

Figure 4: Representación de la imagen con reflejo vertical formato BGR

Pregunta 4: Filtro de color modificado - (5 pts)

Para este ejercicio, se requiere que luego de aplicar el filtro se tenga una imagen con tonalidad medio violeta. Para obtener esta tonalidad se tienen que modificar los valores del píxel en base a los valores originales. Y para lograr dicha tonalidad tienen que utilizar las siguientes fórmulas y reemplazarlos con los valores de la lista tridimensional original:

$$Modificacion\_rojo = 0.298 * original\_rojo + 0.578 * original\_verde + 0.132 * original\_azul$$

$$Modificacion\_verde = 0.254 * original\_rojo + 0.495 * original\_verde + 0.111 * original\_azul$$

$$Modificacion\_azul = 0.177 * original\_rojo + 0.343 * original\_verde + 0.74 * original\_azul$$

Para que el filtro sea válido, deben de **redondear** los valores obtenidos a un entero utilizando la función `round`. Tengan en consideración que si el valor es mayor a 255 (valor máximo en RGB), saldrá un error. En otras palabras, si esos valores llegan a ser de la siguiente forma:

$$R > 255, G > 255, V > 255 \tag{3}$$

van a estar pintando un pixel con un color que no existe ya que el formato RGB se encuentra en un rango [0, 255]. Por esa razón, se tienen que asegurar no pasar ese valor máximo que es 255.

Figure 5: Representación de la imagen con el filtro modificado

Apéndices

A. Ejemplos Gradescope

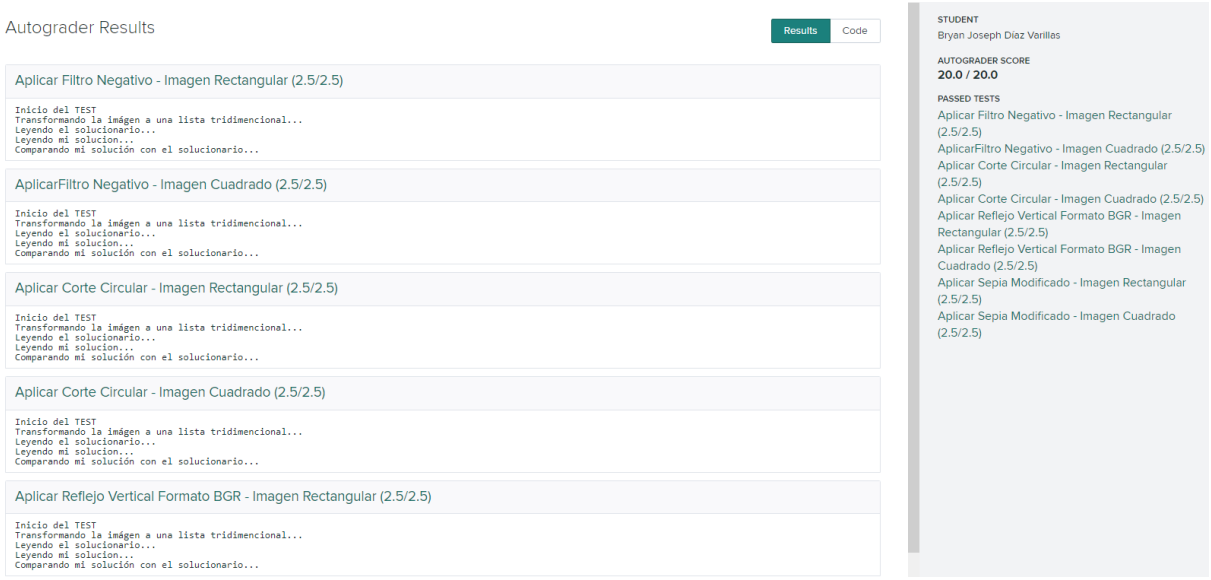

Figure 6: Casos de prueba correctos en Gradescope.

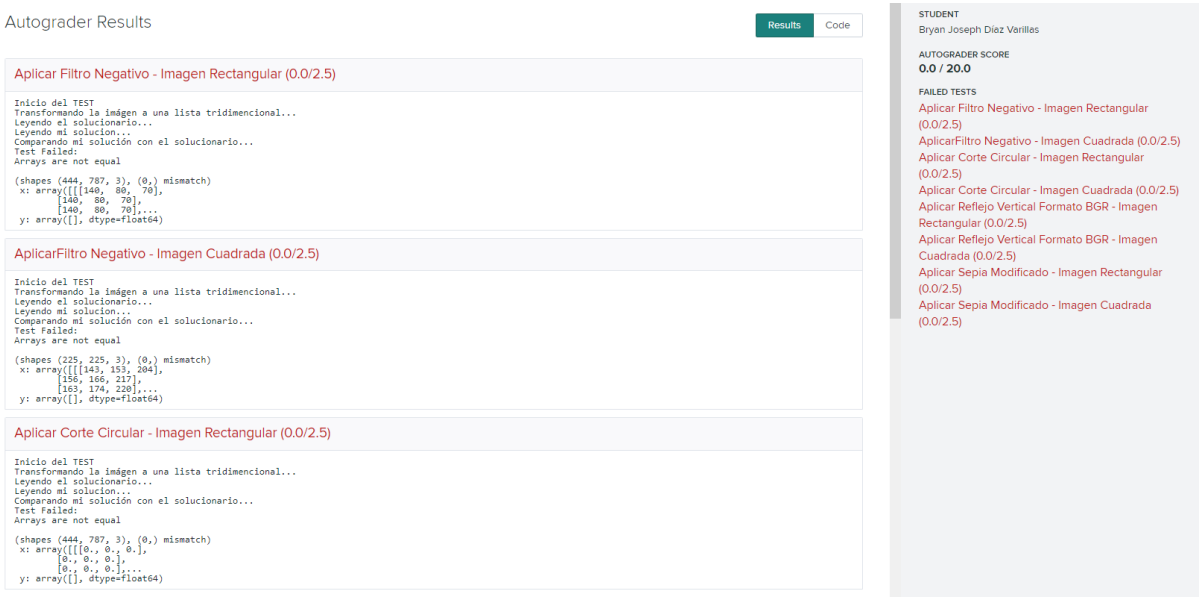

Figure 7: Entrega incorrecta en Gradescope.

B. Template

```python
import imageio
import numpy as np

def leer_imagen(ruta):
    """
    La función leer_imagen recibe un string con la ruta
    de una imagen en formato BMP y retorna una lista de
    tres dimensiones con el mapa de bits de la imagen.
    Asimismo, convertimos la lista de numpy a una lista
    común y corriente.
    """
    np_array = np.array(imageio.imread(ruta), dtype='int')
    # noinspection PyTypeChecker
    lista_3d = np_array.tolist()
    return lista_3d


def guardar_imagen(ruta, lista_3d):
    """
    La función guardar_imagen recibe una lista de 3
    dimensiones con el mapa de bits de la imagen
    y retorna la imagen en formato bmp.
    """
    return imageio.imwrite(ruta, np.array(lista_3d, dtype='uint8'))


class Solution:

    def aplicar_negativo(self, lista_3d=leer_imagen('playa (1).bmp')):
        result = list()
        # SU SOLUCION EMPIEZA AQUI


        # SU SOLUCION TERMINA AQUI
        return result

    def aplicar_corte_circular(self, lista_3d=leer_imagen('playa (1).bmp
    ')):
        result = list()
        # SU SOLUCION EMPIEZA AQUI


        # SU SOLUCION TERMINA AQUI
        return result

```

```
45      def aplicar_reflejo_vertical_BGR(self, lista_3d=leer_imagen('playa
    (1).bmp')):
46          result = list()
47          # SU SOLUCION EMPIEZA AQUI
48
49
50          # SU SOLUCION TERMINA AQUI
51          return result
52
53      def aplicar_sepia_modificado(self, lista_3d=leer_imagen('playa (1).
    bmp')):
54          result = list()
55          # SU SOLUCION EMPIEZA AQUI
56
57
58
59          # SU SOLUCION TERMINA AQUI
60          return result
61
62  if __name__ == '__main__':
63      s = Solution()
64      print("Pregunta #1:")
65      guardar_imagen('negativo.bmp',s.aplicar_negativo())
66      print("Pregunta #2:")
67      guardar_imagen('corte_circular.bmp',s.aplicar_corte_circular())
68      print("Pregunta #3:")
69      guardar_imagen('reflejo_BGR.bmp',s.aplicar_reflejo_vertical_BGR())
70      print("Pregunta #4:")
71      guardar_imagen('sepia_mod.bmp',s.aplicar_sepia_modificado())
```

Listing 1: Template `solution.py`.