```
from matplotlib.image import imread
from helper_functions import *

#----------------------FILL IN THE FOLDER WHERE YOUR IMAGE EXISTS--------
------------------
datafolder = "R:/pythonBTH/assgn1code/images/"
imgpath = datafolder + "6.jpg"
#---------------------------------------STARTER CODE---------------------
-------------------
# Convert the color image to grayscale and returns the grayscale pixels
pixel_values = read_colorimg(imgpath)
# The returned pixel values INCLUDE 2 boundary rows and 2 boundary colns.
Therefore,
numb_rows = len(pixel_values) - 2
numb_colns = len(pixel_values[0]) - 2
#
#---------------------------------------WRITE YOUR CODE HERE-------------
--------------------------
# Create a data structure to store updated pixel information

new_pixel_values = [[0]*numb_colns for i in range(numb_rows)]

# Define the 3 x 3 mask as a tuple of tuples
mask = ((-1,-1,-1), (-1,8,-1), (-1,-1,-1))

# Implement a function to slice a part from the image as a 2D list
def get_slice_2d_list(pixl_Vals,Row_loc,Colmn_loc):
      #slice_List = []
      #for pixel in pixl_Vals[Row_loc-1:Row_loc+2]:
      #     slice_List.append(pixel[Colmn_loc-1:Colmn_loc+2])
      #return slice_List
#-----------------------------------------------------------------

      # list slicing to extract neighbor pixels
      # list comprehension for 2d list neighboring pixels.
      return [r[Colmn_loc-1:Colmn_loc+2]for r in pixl_Vals[Row_loc-
1:Row_loc+2]]



# Implement a function to flatten a 2D list or a 2D tuple.
def flatten(input_slice_List):
      # flattened_list = []
      # for sublist in slice_List:
      #     for item in sublist:
      #           flattened_list.append(item)
#-----------------------------------------------------------------

    #list comprehension to create 1D list from 2D list.
      flatten_list = [item_val for sublist in input_slice_List for
item_val in sublist]
      return flatten_list
```

```
 # For each of the pixel values, excluding the boundary values
 # Create little local 3x3 box using list slicing
for rpixel in range(1,numb_rows+1):
    for cpixel in range(1,numb_colns+1):
                #getting neighbor pixels
        edgeMatrix_pixls = get_slice_2d_list(pixel_values,rpixel,cpixel)
        flat_list = flatten(edgeMatrix_pixls)
        finalMask = flatten(mask)
                # Apply the mask
        edgeMulti_Result = map(lambda k1,k2: k1 * k2, flat_list,
finalMask)
                #Sum all the multiplied values and set the new pixel
value
        new_pixel_values[rpixel-1][cpixel-1]=sum(edgeMulti_Result)

#
#---------------------------------------END YOUR CODE HERE---------------
-------------------------
# Verify your result
verify_result(pixel_values, new_pixel_values, mask)
# View the original image and the edges of the image
view_images(imgpath, new_pixel_values)
```