```
 1
 2 # (1)  Ask the user for an input grayscale image
   and display the input image in the first window.
 3 # (2)  Find the lowest pixel value and the highest
   pixel value in the grayscale image.
 4 # (3)  Make a look-up table to convert the  lowest
   gray value to blue and  the  highest  gray value to
    red.
 5 #      The other gray values should be mapped to
   rainbow colors by the method explained in the
   lecture.
 6 # (4)  Using  OpenCV  functions,  draw  a cross  in
     a circle  to  indicate the pixel  of the  highest
     gray  value.
 7 #      Draw the cross and circle with white. If
   multiple pixels share the same highest gray value,
   place the
 8 #      cross and circle at the center of gravity of
    these pixels.  Figure 4 shows a sample input image
    and
 9 #      output image.
10 # (5)  Save the final color image as input-filename
   -color.png and display the file in the second
   window.
11 import cv2
12 import numpy as np
13 import os
14
15 # User input feature
16 user_input = input("Please name your input color
   file: ")
17 file_directory = os.getcwd()
18 image_location = os.path.join(file_directory,
   user_input)
19 if os.path.exists(image_location):
20     print(f"Your '{user_input}' image loaded
   successfully.")
21
22     grey_image = cv2.imread(user_input, cv2.
   IMREAD_GRAYSCALE)
23     cv2.imshow(f"'{user_input}'", grey_image)
```

```
24      cv2.waitKey(0)
25
26      highest_pixel_value = np.max(grey_image)
27      lowest_pixel_value = np.min(grey_image)
28
29      # color conversion process
30      color_image = np.zeros((grey_image.shape[0],
    grey_image.shape[1], 3), dtype=np.uint8)
31      for i in range(grey_image.shape[0]):
32          for j in range(grey_image.shape[1]):
33              grey_pixel_value = grey_image [i, j]
34              RGB = np.zeros(3, np.uint8)
35
36              if grey_pixel_value <=
    lowest_pixel_value + (highest_pixel_value -
    lowest_pixel_value) / 4:
37                  RGB[0] = 255
38                  RGB[1] = int(255 * (
    grey_pixel_value -lowest_pixel_value) / ((
    highest_pixel_value - lowest_pixel_value) / 4))
39                  RGB[2] = 0
40
41              elif grey_pixel_value <=
    lowest_pixel_value + (highest_pixel_value -
    lowest_pixel_value) / 2:
42                  RGB[0] = int(255 - 255 * (
    grey_pixel_value - lowest_pixel_value - (
    highest_pixel_value - lowest_pixel_value) / 4) / ((
    highest_pixel_value - lowest_pixel_value) / 4))
43                  RGB[1] = 255
44                  RGB[2] = 0
45
46              elif grey_pixel_value <=
    lowest_pixel_value + 3 * (highest_pixel_value -
    lowest_pixel_value) / 4:
47                  RGB[0] = 0
48                  RGB[1] = 255
49                  RGB[2] = int(255 * (
    grey_pixel_value - lowest_pixel_value - 2 * (
    highest_pixel_value - lowest_pixel_value) / 4) / ((
    highest_pixel_value - lowest_pixel_value) / 4))
```

```
50              else:
51                  RGB[0] = 0
52                  RGB[1] = int(255 - 255 * (
   grey_pixel_value - lowest_pixel_value - 3 * (
   highest_pixel_value - lowest_pixel_value) / 4) / ((
   highest_pixel_value - lowest_pixel_value) / 4))
53                  RGB[2] = 255
54
55              if RGB[1] < 0:
56                  RGB[1] = 0
57              elif RGB[1] > 255:
58                  RGB[1] = 255
59
60              color_image[i, j] = RGB
61
62      # Finding the highest grey value
63      highest_pixel_coordinate = np.argwhere(
   grey_image == highest_pixel_value)
64      center_y, center_x = highest_pixel_coordinate[0
   ]
65
66      cv2.circle(color_image, (center_x, center_y),
   20, (255, 255, 255), 3)  # White circle
67
68      cross_size = 30
69      cv2.line(color_image, (center_x - cross_size,
   center_y), (center_x + cross_size, center_y), (255
   , 255, 255), 3)
70      cv2.line(color_image, (center_x, center_y -
   cross_size), (center_x, center_y + cross_size), (
   255, 255, 255), 3)
71
72      cv2.imshow(f"'{user_input}'", color_image)
73      cv2.waitKey(0)
74
75      # Saving the output image
76      output_image = user_input.split('.')[0] + '-
   color.' + user_input.split('.')[-1]
77      cv2.imwrite( output_image, color_image)
78
79      print(f"The highest pixel value for '{
```

```
79 user_input}' is: {highest_pixel_value}")
80     print(f"The lowest pixel value for '{
   user_input}' is: {lowest_pixel_value}")
81
82 else:
83     print(f"Error: unable to load your input image
   .\nPlease make sure '{user_input}' is in the
   correct directory.")
84     exit()
85
86
```