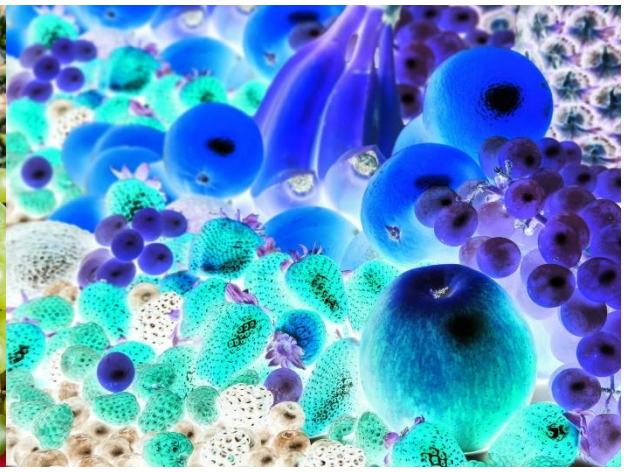
1. Write a program to convert the image “fruit.jpg” to its complementary colors and display it on the screen.

```
import numpy as np
import cv2
import os

image = cv2.imread("fruit.jpg")

complementary_image = 255 - image

cv2.imshow('Original Image', image)
cv2.imshow('Complementary Image', complementary_image)

cv2.waitKey(0)
cv2.destroyAllWindows()

output_folder = "./complementary"
if not os.path.exists(output_folder):
    os.makedirs(output_folder)
cv2.imwrite(os.path.join(output_folder,"complementary_fruit.jpg"),
complementary_image)
```

ภาพต้นฉบับ

ภาพ complementary colors



อธิบายเพิ่มเติม : การทำ complementary colors เป็นการแสดงผลสีที่เป็นลักษณะสีตรงข้ามกับภาพต้นฉบับ ใน GBR color space การทำ complementary สามารถทำได้โดยการนำ 255 ลบด้วย ค่า G, B, R แต่ละ pixels ของแต่ละ planes แล้วนำมา merge รวมกัน

2. Use color slicing to segment the oranges from the image “oranges.jpg” by keeping the colors of the oranges to be same but changing the colors of other fruits to blue color (0,0,0.5). Find the most suitable range of orange color using your judgement.

```
import numpy as np
import cv2
import os

def RGB_TO_HSI(img):
    # Normalize to [0, 1]
    img = img.astype(np.float32) / 255.0

    with np.errstate(divide='ignore', invalid='ignore'):
        b, g, r = cv2.split(img)

        # Calculate Intensity
        intensity = np.divide(b + g + r, 3)

        # Calculate Saturation
        min_rgb = np.minimum(np.minimum(r, g), b)
        saturation = 1 - 3 * np.divide(min_rgb, r + g + b + 1e-6)  # Add a small
epsilon to avoid division by zero

        # Calculate Hue
        numerator = ((r - g) + (r - b)) / 2.0
        denominator = np.sqrt((r - g)**2 + (r - b) * (g - b))

        # Avoid division by zero
        denominator = np.where(denominator == 0, 1e-6, denominator)
        hue = np.arccos(numerator / denominator)

        # Adjust hue range to be between 0 and 2*pi
        hue[b > g] = 2 * np.pi - hue[b > g]

        return hue, saturation, intensity


def segment_oranges(image, hue_range):

    # Convert RGB to HSI
    hue, saturation, intensity = RGB_TO_HSI(image)

    # Create a mask based on hue, saturation, and intensity
    hue_mask = (hue >= hue_range[0]) & (hue <= hue_range[1])
```

```
    # Combine the masks
    mask = hue_mask

    # Change non-orange parts to blue
    image[mask == False] = [255, 0, 0]  # BGR format for blue

    return image

lower_orange_hue = 0.1
upper_orange_hue = 1


image = cv2.imread("oranges.jpg")
segmented_image = segment_oranges(image, (lower_orange_hue, upper_orange_hue))

cv2.imshow('Segmented Image', segmented_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

output_folder = "./color_slicing"
if not os.path.exists(output_folder):
    os.makedirs(output_folder)
cv2.imwrite(os.path.join(output_folder,"segmented_oranges.jpg"), segmented_image)
```

อธิบายเพิ่มเติม : การทำ color slicing เป็นการเอาเฉพาะส่วนสีที่ต้องการเช่นในภาพนี้จะเป็นสีส้มของผลส้ม โดยหลักการสามารถทำจาก RGB ก็ได้แต่ส่วนตัวคิดว่าจะยากกว่า จึงแปลงภาพเป็น HSI ก่อน แล้วค่อยปรับแค่ค่า H ที่เป็นค่าของสี โดยสีแดงจะมีค่า 0 deg ประมาณ 0 rad, สีเขียวจะมีค่า 120 deg ประมาณ 2.1 rad และสีน้ำเงินจะมีค่า 240 deg ประมาณ 4.2 rad การลองแทนค่าจึงเลือกช่วง 0.1-1 ในการตั้งเป็นขอบที่กำหนดว่าเป็นส่วนของส้ม หมายความว่า pixels ที่มีค่า H อยู่ระหว่าง 0.1 ถึง 1 จะแสดงสีคงเดิมคือสีส้ม ส่วนที่ไม่ใช่ mask จะ filter ออกและปรับเป็นสีน้ำเงิน