Ahmed Asaad
G1916043

Description of the Algorithm

In the first part of the code, the algorithm starts by reading the image which we want to extract the number of objects from. The image should ne read as a grayscale image then to be converted to a binary image. For a later purpose, the binary image mask in inverted. This is to make the objects as a foreground instead of background. The morphological operation "opening" is used to enhance the nature of objects in the image to make every object as one continuous item. It uses a 5X5 kernel. This is done to avoid the wrong reading of number the object if it is discrete.

Then the connected components algorithm is used to label the image. It makes all connected pixels as one component. This woks as segmentation and identifying of the separated and labeled objects. In this case I used four connectivity pixels.

Then The number of objects is printed out using "print" command. The variable "num_labels" assigned the value of labels extracted from connectedComponents algorithms including the background label. So I subtracted one from the value to exclude the background label.

The second part of the code used to give colors to the labels or parts of the image. After that it is displayed using the cv2.imshow command.

The flowchart:

```
┌─────────────────────────────┐
│  Read the image as grayscale │
└─────────────────────────────┘
              ↓
┌─────────────────────────────┐
│    Convert grayscale to      │
└─────────────────────────────┘
              ↓
┌─────────────────────────────┐
│  Inversing the mask of binary │
│            image             │
└─────────────────────────────┘
              ↓
┌─────────────────────────────┐
│     modifying image using    │
│   morphological operation    │
└─────────────────────────────┘
              ↓
┌─────────────────────────────┐
│  Labling the image (identifying │
│      segments of image)      │
└─────────────────────────────┘
              ↓
┌─────────────────────────────┐
│  Printing the number of labels │
└─────────────────────────────┘
```