In [2]:

```
#Libraries
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
```

Task 1: Thresholding-Based Segmentation Scenario- You have a grayscale medical X-ray image of a bone fracture. The area of interest (the fracture) is significantly darker than the surrounding bone. Perform thresholding-based segmentation to isolate the fracture.

In [14]:

```
image1 = cv2.imread('q1.png', cv2.COLOR_BGR2GRAY)
plt.imshow(image1)
```

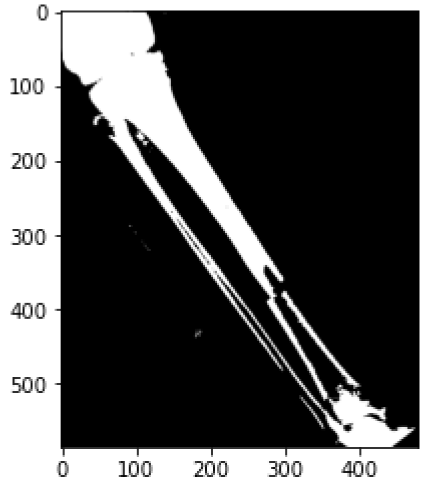
Out[14]: <matplotlib.image.AxesImage at 0x27be791b3a0>

In [21]:

```
_, binarymask = cv2.threshold(image1, 128, 255, cv2.THRESH_BINARY)
```

In [22]:
```
plt.imshow(binarymask)
```

Out[22]: <matplotlib.image.AxesImage at 0x27be7a61460>

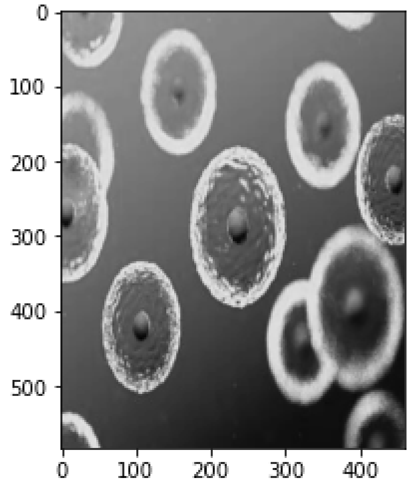


Task 2: Region Growing Intensity-Based Segmentation Scenario- You have a microscopic image of cells. Choose a seed point in one of the cells, and perform region growing- based segmentation to identify and separate that cell from the rest.

In [5]:

```python
image2 = cv2.imread('q2.png',cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(image2, cv2.COLOR_RGB2GRAY)
plt.imshow(gray, cmap = 'gray')
```

Out[5]: <matplotlib.image.AxesImage at 0x2291ced99d0>



In [19]:

```python
image = cv2.imread("q2.png", cv2.IMREAD_GRAYSCALE)
```

In [11]:

```python
seedpoint = (50, 75)
threshold = 60
```

In [20]:

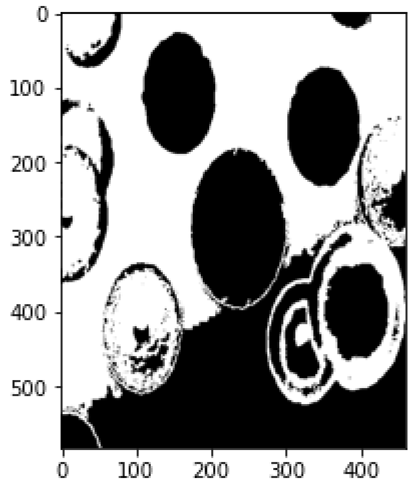
```python
def regionGrowing(image, seed, threshold):
    mask = np.zeros_like(image, dtype = np.uint8)
    stack = [seed]
    seed_intensity = image[seed]

    while stack:
        x, y = stack.pop()
        if x < 0 or x >= image.shape[0] or y < 0 or y >= image.shape[1]:
            continue

        if mask[x, y] == 0:
            if abs(int(image[x, y]) - int(seed_intensity)) < threshold:
                mask[x, y] = 255
                stack.extend([(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)])
    return mask
```

In [21]:

```python
segmentedImage = regionGrowing(image, seedpoint, threshold)
```

In [23]:

```python
plt.imshow(segmentedImage, cmap = 'gray')
plt.show()
```



Task 3: Watershed Segmentation Scenario- You have an image of overlapping coins on a table. Perform watershed segmentation to separate and count the individual coins.

In [118]:
```
image3 = cv2.imread('q3.png')
gray = cv2.cvtColor(image3, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(image3, cv2.COLOR_BGR2RGB))
```

Out[118]: <matplotlib.image.AxesImage at 0x27beb902760>

In [119]:
```
_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

In [120]:
```
kernel = np.ones((5, 5), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations = 2)
```

In [147]:
```
sure_bg = cv2.dilate(opening, kernel, iterations = 3)
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist_transform, 0.9 * dist_transform.max(), 255, 0)
```

In [148]:

```
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)
```

In [149]:

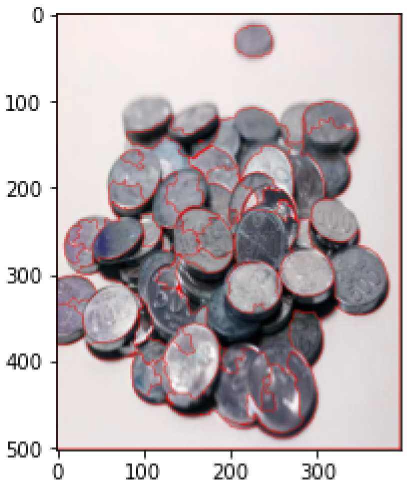
```
_, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0
```

In [150]:

```
cv2.watershed(image3, markers)
image3[markers == -1] = [255, 0, 0]
```

In [151]:

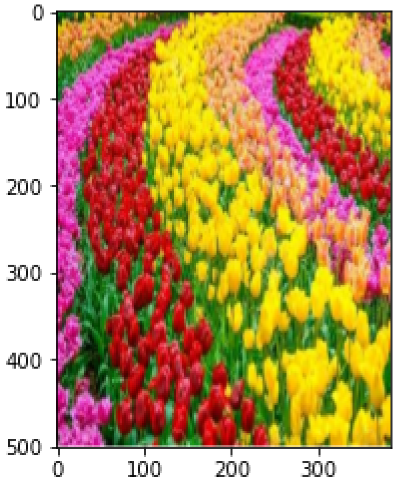
```
plt.imshow(image3)
plt.axis('on')
plt.show()
```



Task 4: Cluster-Based Segmentation Scenario- You have an image of colorful flowers in a garden. Perform cluster-based segmentation to separate different types of flowers based on color.

In [152]:
```
image4 = cv2.imread('q4.png')
plt.imshow(cv2.cvtColor(image4, cv2.COLOR_BGR2RGB))
```

Out[152]: <matplotlib.image.AxesImage at 0x27bebc31ac0>



In [153]:
```
pixel_values = image4.reshape((-1, 3))
pixel_values = np.float32(pixel_values)
```

In [154]:
```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
```

In [167]:
```
K = 3
```

In [168]:
```
_, labels, centers = cv2.kmeans(pixel_values, K, None, criteria, 50, cv2.KMEANS_RANDOM_CENTERS)
```

In [169]:
```
centers = np.uint8(centers)
```

In [170]:

```
segmented_image = centers[labels.flatten()]
segmented_image = segmented_image.reshape(image4.shape)
```

In [171]:

```
plt.imshow(cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB))
```

Out[171]:

```
<matplotlib.image.AxesImage at 0x27bec15db80>
```