```python
#%% Part 1
import numpy as np
import cv2

#Setting up the window with 20 by 20 dots
Spacing_Para = 20 # Spacing_Para represents the space between each 2 dots.

img = np.zeros((420,420,3),dtype="uint8") # Defining the size of the window.

#The following 'for loop' is setting up the 20 by 20 dots in the window.
for i in range (20):
    for j in range (20):
        cv2.rectangle(img,(Spacing_Para*i+9, Spacing_Para*j+9),(Spacing_Para*i+16, Spacing_Para*j+16),(255,255,255),-1)
        # Dimension for each dot is 7 by 7.


ix,iy=-1,-1 # Just initial values with no meaning and they will be redefined later.
def I_wanna_get_hired(event,x,y,flags,param):
#Building a function called 'I_wanna_get_hired':
#(1) Drawing the user-defined circle;
#(2) Highlighting the dots on circle;
#(3) Drawing inner and outer circles based on the distances btw each highlighted dot and the center of the circle.

    global ix,iy # Setting these variable as global, so once the loops are fininshed, the updated variables won't be erased.

    if event == cv2.EVENT_LBUTTONDOWN: # If the user press the left button and hold it,
        ix,iy = x,y                    # recording the cursor position when the user is holding the left button.

    elif event == cv2.EVENT_LBUTTONUP: # When the user release the left button, excecute the command in 'elif' section.

        r = pow((ix-x)**2 + (iy-y)**2, 0.5) # Radius of the circle, defined by user's mouse button action (hold and release).
                                            # The cursor's coordinate according to 'holding left button' is ix,iy
                                            # The cursor's coordinate according to 'releasing left button' is x,y
                                            # The radius is the distance btw above 2 coordinates.

        cv2.circle(img,(ix,iy),round(r),(255,0,0),2) # Drawing the circle

        D = [] # Later on we will fill this empty set with eaac highlighted dot's distance to the center of user-defined circle.
        for i in range (20):
            for j in range (20):
                d = pow((Spacing_Para*i+(9+16)/2 - ix)**2 + (Spacing_Para*j+(9+16)/2 -iy)**2,0.5)
```

```
                # d represents the distance between the geometric center of each dot to the center of the user-defined circle.

                if d >= r - 3.5* (pow(3.5,0.5)) and d <= r + 3.5*(pow(3.5,0.5)):
                # The tolerance I used here is half of the dignaol line of each dot, (7*sqrt(2))/2.

                    cv2.rectangle(img,(Spacing_Para*i+9, Spacing_Para*j+9),(Spacing_Para*i+16, Spacing_Para*j+16),(0,255,0),-1)
                    # The dots satisfing the if statement regarding d will be in green color.

                    D.append(d)
                    # Here we store distances btw all qualified dots's geometric center to the user-defined circle's center.

        else: # This section started with 'else:' will be executed only when the 'for loop' above finished with no break.
            cv2.circle(img,(ix,iy),round(max(D)),(0,0,255),2) # Drawing the outer circle based on the maximum distance in set D.
            cv2.circle(img,(ix,iy),round(min(D)),(0,0,255),2) # Drawing the inner circle based on the minimum distance in set D.

cv2.namedWindow('image')
cv2.setMouseCallback('image',I_wanna_get_hired) # Run the 'I_wanna_get_hired' function I just built.
while(1):
    cv2.imshow('image',img)
    if cv2.waitKey(20) & 0xFF == 27:
        break
cv2.destroyAllWindows()
```