```
import skimage.io as io
import numpy as np
from skimage.color import rgb2gray
from skimage.filters import sobel

# This function is provided to you. You will need to call it.
# You should not need to modify it.
def seedfill(im, seed_row, seed_col, fill_color, bckg):
    """
    im: The image on which to perform the seedfill algorithm
    seed_row and seed_col: position of the seed pixel
    fill_color: Color for the fill
    bckg: Color of the background, to be filled
    Returns: Number of pixels filled
    Behavior: Modifies image by performing seedfill
    """
    size=0  # keep track of patch size
    n_row, n_col = im.shape
    front={(seed_row,seed_col)}  # initial front
    while len(front)>0:
        r, c = front.pop()  # remove an element from front
        if im[r, c]==bckg:
            im[r, c]=fill_color  # color the pixel
            size+=1
            # look at all neighbors
            for i in range(max(0,r-1), min(n_row,r+2)):
                for j in range(max(0,c-1),min(n_col,c+2)):
                    # if background, add to front
                    if im[i,j]==bckg and\
                       (i,j) not in front:
                        front.add((i,j))
    return size


# QUESTION 1
def detect_edges(image_color):
    """
    Args:
        image_color: the original color image read from an image_file
    Returns:
        image_graytone: a grayscale image converted from the original
image
        image_sobel: a new image of detected edges
    """
    # WRITE YOUR CODE HERE



# QUESTION 2
```

```
def binarize_edges(image_sobel1):
    """
    Args:
        image_sobel1: an ndarray as the initial edge detected image
from Q1
    Returns:
        image_sobel2: a new image where the pixels whose value grear
or equal to
        0.05 are set to 1; othewise to 0
    """
    # WRITE YOUR CODE HERE




# QUESTION 3
def cleanup(image_graytone, image_sobel2):
    """
    Args:
        image_graytone: the grayscale image from Q1
        image_sobel2: the image from Q2 where pixel value is either 0
or 1
    Returns:
        image_sobel3: a modified version of image_sobel2, where any
white pixel at
        position (r,c) is replaced by a black pixel if the value of
pixel (r,c)
        or of any of the 8 surrounding pixels in image_graytone image
is below 0.5.
    """
    # WRITE YOUR CODE HERE




# QUESTION 4
def fill_cells(image_sobel3):
    """
    Args:
        edge_image: A black-and-white image, with black background and
                    white edges
    Returns:
        filled_image: A new image where each close region is filled
with
                    a different grayscale value
    """
    # WRITE YOUR CODE HERE
```

```
# QUESTION 5
def classify_cells(image_graytone, filled_image, \
                   min_size=1000, max_size=5000, \
                   infected_grayscale=0.5,
min_infected_percentage=0.02):
    """
    Args:
        image_graytone: The graytone image from Q1
        filled_image: A graytone image, with each closed region
colored
                        with a different grayscal value from Q4
        min_size, max_size:
            The min and max size of a region to be called a cell
        infected_grayscale:
            Maximum grayscale value for a pixel to be called infected
        min_infected_percentage:
            Smallest fraction of dark pixels needed to call a cell
infected
    Returns: A tuple of two sets, containing the grayscale values of
cells
             that are infected and not infected
    """
    # WRITE YOUR CODE HERE



# QUESTION 6
def annotate_image(color_image, filled_image, infected, not_infected):
    """
    Args:
        image_color: the original color image read from image_file in
Q1
        filled_image: A graytone image, with each closed region
colored
                        with a different grayscal value in Q4
        infected: A set of graytone values of infected cells
        not_infected: A set of graytone values of non-infcted cells
    Returns: A color image, with infected cells highlighted in red
             and non-infected cells highlighted in green
    """
    # WRITE YOUR CODE HERE



if __name__ == "__main__":  # do not remove this line

    # QUESTION 1 TEST CODE
    # The path of the malaria image file. Default to "malaria-1.jpg"
    # Change it to "malaria-1-small.jpg" for quick debugging
```

```
    image_file = "malaria-1.jpg"
    # image_file = "malaria-1-small.jpg"

    image_color = io.imread(image_file)
    image_graytone,image_sobel1 = detect_edges(image_color)
    io.imsave("Q1_gray_sobel.jpg",image_sobel1)

    # QUESTION 2 TEST CODE
    image_sobel2 = binarize_edges(image_sobel1)
    io.imsave("Q2_gray_sobel_T005.jpg",image_sobel2)

    # QUESTION 3 TEST CODE
    image_sobel3 = cleanup(image_graytone, image_sobel2)
    io.imsave("Q3_gray_sobel_T005_cleanup.jpg",image_sobel3)

    # QUESTION 4 TEST CODE
    image_filled=fill_cells(image_sobel3)
    io.imsave("Q4_gray_sobel_T005_cleanup_filled.jpg",image_filled)

    # QUESTION 5 TEST CODE
    infected, not_infected = classify_cells(image_graytone,
image_filled)
    print(infected)
    print(not_infected)

    # QUESTION 6 TEST CODE
    annotated_image = annotate_image(image_color, image_filled,
                                     infected, not_infected)
    io.imsave("Q6_annotated.jpg", annotated_image)
```