```python
from skimage.io import imread
from skimage.exposure import adjust_gamma
from skimage.morphology import dilation, square, remove_small_objects, label
from skimage.segmentation import flood_fill
from utilities import roberts, transform, unsharp_mask
import matplotlib.pyplot as plt
import numpy as np
from random import randint, choice

"""# 1. Read “balloons.jpg” image. Find the outer edges (not the patterns inside) of the air balloons."""
# load the image and use only the green channel
orig_img = imread("balloons.jpg")
img = orig_img[:, :, 2] / 255.0

# Preprocess the image such that balloons are darker and background is uniform color.
img = adjust_gamma(img, gamma=2)
img = unsharp_mask(img)
img = img < 0.12
img = dilation(img, square(3))

# Perform edges detection with Roberts Cross
img = roberts(img)
img = img > 0
bin_img = remove_small_objects(img, 75)


"""2. Count the total number of the balloons"""
labels_img = label(bin_img)
num_bln = np.max(labels_img)


"""# 3. Plot the resulting image from step 1, and as a title of your image, write the total number of the balloons you found in step 2. (No hard coding please) and then save the resulting image as a png."""
fig, ax = plt.subplots(1, 2)
fig.suptitle(f"There are {num_bln:d} balloons in the image.", fontsize=16)
ax[0].set_title("Original image")
ax[0].imshow(orig_img)
ax[1].set_title("Edges image")
ax[1].imshow(bin_img, cmap="gray")
fig.savefig("3. detected_edges.png", dpi=1000)
plt.close()


"""# 4. Choose a random air balloon in your binary image, change the pixels inside to white. Explain how you did that."""
# Pick a balloon based on label
bln = randint(1, num_bln)

# Find all pixels belong to that balloon
rows, cols = np.where(labels_img == bln)

# Find center the ballon
cen_row, cen_col = round(np.average(rows)), round(np.average(cols))

# Use flood_fill algo to fill the ballon
```

```
fill_img = np.copy(bin_img)
fill_img = flood_fill(fill_img, (cen_row, cen_col), 1)

# Save filled image
fig = plt.figure()
plt.imshow(fill_img, cmap="gray")
fig.suptitle(f"Balloons {bln:d} is randomly picked to be fill.")
fig.savefig("4. fill_a_balloons.png", dpi=1000)
plt.close()


"""# 6. Move the balloon 20 pixels in any direction of 45-degree angle. Explain how you did that."""
# Find all pixels belong to a ballon
# Fill the ballon
fill_img = flood_fill(labels_img, (cen_row, cen_col), bln)
rows, cols = np.where(fill_img == bln)

# Create a new transform image based on the original image
tf_img = np.copy(orig_img)

# Set transform parameters
tf_row = choice([-20, 20])
tf_col = choice([-20, 0, 20])
angle = 45  # in degree

# Set pixels of the balloon's orignal location to white
for row, col in zip(rows, cols):
    tf_img[row, col, :] = 255

# Copy pixels from the balloon's orignal location to the new location
for row, col in zip(rows, cols):
    new_row, new_col = transform(
        (row, col), (cen_row, cen_col), (tf_row, tf_col), angle
    )
    if 0 <= new_row < tf_img.shape[0] and 0 <= new_col < tf_img.shape[1]:
        tf_img[new_row, new_col, :] = orig_img[row, col, :]

# Save transform image
fig = plt.figure()
plt.imshow(tf_img)
fig.suptitle(
    f"Balloons {bln:d} shifts vertically by {tf_row:d} pixels, horizontally by {tf_col:d} pixels, and rotate by {angle:d}\N{DEGREE SIGN}",
    fontsize=10,
)
fig.savefig("6. transform_balloons.png", dpi=1000)
plt.close()


"""# 7. Rotate the air balloon 60 degrees clockwise after step 6."""
# Find all pixels belong to a ballon
# Fill the ballon
fill_img = flood_fill(labels_img, (cen_row, cen_col), bln)
rows, cols = np.where(fill_img == bln)

# Create a new transform image based on the original image
tf_img = np.copy(orig_img)
```

```python

# Set transform parameters
angle = 45 + 60  # in degree

# Set pixels of the balloon's orignal location to white
for row, col in zip(rows, cols):
    tf_img[row, col, :] = 255

# Copy pixels from the balloon's orignal location to the new location
for row, col in zip(rows, cols):
    new_row, new_col = transform(
        (row, col), (cen_row, cen_col), (tf_row, tf_col), angle
    )
    if 0 <= new_row < tf_img.shape[0] and 0 <= new_col < tf_img.shape[1]:
        tf_img[new_row, new_col, :] = orig_img[row, col, :]

# Save transform image
fig = plt.figure()
plt.imshow(tf_img)
fig.suptitle(
    f"Balloons {bln:d} shifts vertically by {tf_row:d} pixels, horizontally by {tf_col:d} pixels, and rotate by {angle:d}\N{DEGREE SIGN}",
    fontsize=10,
)
fig.savefig("7. rotate_balloons_again.png", dpi=1000)
plt.close()
```