Lab exam:application question:

```
import numpy as np
import random
from PIL import Image, ImageEnhance
from skimage import feature
import matplotlib.pyplot as plt
def evaluate_fitness(params, image):
    alpha, beta, threshold1, threshold2 = params[0], params[1], params[2], params[3]
    enhancer = ImageEnhance.Contrast(image)
    adjusted_image = enhancer.enhance(alpha)
    edges = feature.canny(np.array(adjusted_image), sigma=1)
    edge_density = np.sum(edges) / np.prod(edges.shape)
    return -edge_density
def random_solution(lb, ub):
    return np.random.uniform(lb, ub)
def levy_flight(alpha, dim):
    u = np.random.normal(0, 1, dim)
    v = np.random.normal(0, 1, dim)
    step = u / np.power(np.abs(v), 1 / alpha)
    return step
def cuckoo_search(num_nests, max_iter, lb, ub, alpha, pa, image):
    nests = [random_solution(lb, ub) for _ in range(num_nests)]
    fitness = [evaluate_fitness(nest, image) for nest in nests]
    best_nest = nests[np.argmin(fitness)]
    best_fitness = min(fitness)
    for iteration in range(max_iter):
        cuckoo = best_nest + levy_flight(alpha, len(best_nest))
        cuckoo = np.clip(cuckoo, lb, ub)
        cuckoo_fitness = evaluate_fitness(cuckoo, image)
        random_index = random.choice(range(num_nests))
        if cuckoo_fitness < fitness[random_index]:
            nests[random_index] = cuckoo
            fitness[random_index] = cuckoo_fitness
        best_index = np.argmin(fitness)
        best_nest = nests[best_index]
        best_fitness = fitness[best_index]
        for i in range(num_nests):
            if random.uniform(0, 1) < pa:
                nests[i] = random_solution(lb, ub)
                fitness[i] = evaluate_fitness(nests[i], image)
        print(f"Iteration {iteration + 1}: Best Fitness = {best_fitness}")
    print("Best Solution:", best_nest)
    return best_nest
num_nests = 25
max_iter = 50
dim = 4
lb = np.array([1.0, 0, 50, 100])
ub = np.array([3.0, 100, 150, 200])
alpha = 1.5
pa = 0.25
image = Image.fromarray(np.random.randint(0, 256, (256, 256), dtype=np.uint8)).convert('L')
best_params = cuckoo_search(num_nests, max_iter, lb, ub, alpha, pa, image)
alpha, beta, threshold1, threshold2 = best_params
enhancer = ImageEnhance.Contrast(image)
adjusted_image = enhancer.enhance(alpha)
edges = feature.canny(np.array(adjusted_image), sigma=1)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(np.array(image), cmap='gray')
plt.title("Original Image")
plt.subplot(1, 2, 2)
plt.imshow(edges, cmap='gray')
plt.title("Enhanced Image with Edges")
plt.show()
```

OUTPUT:

```
Iteration 1: Best Fitness = -0.3896484375
Iteration 2: Best Fitness = -0.3896484375
Iteration 3: Best Fitness = -0.3895111083984375
Iteration 4: Best Fitness = -0.3895111083984375
Iteration 5: Best Fitness = -0.3895111083984375
Iteration 6: Best Fitness = -0.3895111083984375
Iteration 7: Best Fitness = -0.3895111083984375
Iteration 8: Best Fitness = -0.3893585205078125
Iteration 9: Best Fitness = -0.3893585205078125
Iteration 10: Best Fitness = -0.3893585205078125
Iteration 11: Best Fitness = -0.389495849609375
Iteration 12: Best Fitness = -0.3893585205078125
Iteration 13: Best Fitness = -0.3896484375
Iteration 14: Best Fitness = -0.3896484375
Iteration 15: Best Fitness = -0.3894805908203125
Iteration 16: Best Fitness = -0.3894805908203125
Iteration 17: Best Fitness = -0.3894805908203125
Iteration 18: Best Fitness = -0.3894805908203125
Iteration 19: Best Fitness = -0.3895263671875
Iteration 20: Best Fitness = -0.3895263671875
Iteration 21: Best Fitness = -0.3894805908203125
Iteration 22: Best Fitness = -0.3894805908203125
Iteration 23: Best Fitness = -0.3893585205078125
Iteration 24: Best Fitness = -0.38946533203125
Iteration 25: Best Fitness = -0.38946533203125
Iteration 26: Best Fitness = -0.38946533203125
Iteration 27: Best Fitness = -0.3896026611328125
Iteration 28: Best Fitness = -0.3896026611328125
Iteration 29: Best Fitness = -0.3896026611328125
Iteration 30: Best Fitness = -0.3896484375
Iteration 31: Best Fitness = -0.3893585205078125
Iteration 32: Best Fitness = -0.3896484375
Iteration 33: Best Fitness = -0.3893585205078125
Iteration 34: Best Fitness = -0.3893585205078125
Iteration 35: Best Fitness = -0.3893585205078125
Iteration 36: Best Fitness = -0.3893585205078125
Iteration 37: Best Fitness = -0.3893585205078125
Iteration 38: Best Fitness = -0.3893585205078125
Iteration 39: Best Fitness = -0.3893585205078125
Iteration 40: Best Fitness = -0.389617919921875
Iteration 41: Best Fitness = -0.389617919921875
Iteration 42: Best Fitness = -0.3896484375
Iteration 43: Best Fitness = -0.3896484375
Iteration 44: Best Fitness = -0.3896484375
Iteration 45: Best Fitness = -0.3896484375
Iteration 46: Best Fitness = -0.3896484375
Iteration 47: Best Fitness = -0.3896484375
```

```
Iteration 48: Best Fitness = -0.3896484375
Iteration 49: Best Fitness = -0.3896484375
Iteration 50: Best Fitness = -0.3896484375
Best Solution: [  2.99488267  29.52532804 114.33637408 188.13739757]
```

Sample image: