```python
import random
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('TkAgg')

n = 16 # Number of Queens
p = 100  # Number of Population

current_generation = []  # Current Generation
new_generation = []  # New Generation


def RandomPopulationGeneration(num_rows, num_queens):
    """
    :param num_rows: Rows of the board.
    :param num_queens: Number of queens for the problem.
    :return: List of the random board generated.
    """
    list_of_generation = []
    for i in range(num_rows):
        gene = []
        for j in range(num_queens):
            gene.append(random.randint(1, n))
        gene.append(0)
        list_of_generation.append(gene)
    return list_of_generation


def FitnessSurvival(population):
    """

    :param population: List of the population of the board generated.
    :return: Best fitting population after swap is made.
    """
    i = 0
    attacking = 0
    while i < len(population):
        j = 0
        attacking = 0
        while j < n:
            l = j + 1

            while l < n:
                if population[i][j] == population[i][l]:
                    attacking += 1
                if abs(j - l) == abs(population[i][j] - population[i][l]):
                    attacking += 1
                l += 1
            j += 1
        population[i][len(population[j]) - 1] = attacking
        i += 1

    for i in range(len(population)):
        minimum = i
        for j in range(i, len(population)):
            if population[j][n] < population[minimum][n]:
                minimum = j

        temp = population[i]
        population[i] = population[minimum]
        population[minimum] = temp
    return population


def CrossOver(list_of_generation):
    """
```

```
    :param list_of_generation: List of the generations for crossover
    :return: Completed cross over  list
    """
    for i in range(0, len(list_of_generation), 2):
        z = 0
        new_child1 = []
        new_child2 = []
        while z < n:
            if (z < n // 2):
                new_child1.append(list_of_generation[i][z])
                new_child2.append(list_of_generation[i + 1][z])
            else:
                new_child1.append(list_of_generation[i + 1][z])
                new_child2.append(list_of_generation[i][z])
            z += 1
        new_child1.append(0)
        new_child2.append(0)
        list_of_generation.append(new_child1)
        list_of_generation.append(new_child2)
    return list_of_generation


def Mutation(list_of_generation):
    """

    :param list_of_generation: List for mutation function
    :return: Lis of mutated population.
    """
    list_of_mutation = []
    i = 0
    while i < p // 2:
        new_rand = random.randint(p // 2, p - 1)
        if new_rand not in list_of_mutation:
            list_of_mutation.append(new_rand)
            list_of_generation[new_rand][random.randint(0, n - 1)] = random.randint(1, n - 1)
            i += 1
    return list_of_generation


def ShowResults(response):
    """

    :param response: Plot the queens position using matplotblib
    :return: Show plot.
    """
    l = len(response)
    plt.figure(figsize=(6, 6))
    plt.scatter([x + 1 for x in range(l - 1)], response[:l - 1])
    for i in range(l):
        plt.plot([0.5, l - 0.5], [i + 0.5, i + 0.5], color="k")
        plt.plot([i + 0.5, i + 0.5], [0.5, l - 0.5], color="k")
    plt.show()


# Call the driver program.
current_generation = RandomPopulationGeneration(p, n)
current_generation = FitnessSurvival(current_generation)
epoch = 1
while True:
    print("------------------------------------------------------")
    print("Epoch ", epoch)
    current_generation = current_generation[0:p // 2]
    new_generation = CrossOver(current_generation)
    new_generation = Mutation(new_generation)
    current_generation = new_generation
    current_generation = FitnessSurvival(current_generation)
    if current_generation[0][n] == 0:
```

```
        print("Solution Found: ", current_generation[0])
        ShowResults(current_generation[0])
        break
    else:
        print("Best Solution: ", current_generation[0])
    epoch += 1
```