A-I ASSIGNMENT

Y. Gopal Krishna
AP21110010189
CSE-C

Code:

```
import random

cities = [(0, 0), (1, 2), (2, 4), (3, 1), (4, 3)]

calculate_distance = lambda city1, city2: ((city2[0] - city1[0]) ** 2 + (city2[1] - city1[1]) ** 2) ** 0.5

num_cities, population_size, num_generations, mutation_rate = len(cities), 50, 100, 0.01
population = [random.sample(range(num_cities), num_cities) for _ in range(population_size)]

calculate_total_distance = lambda tour: sum(calculate_distance(cities[tour[i]], cities[tour[(i + 1) % num_cities]]) for i in range(num_cities))

for _ in range(num_generations):
   fitness_scores = [1 / calculate_total_distance(tour) for tour in population]
   selected_indices = random.choices(range(population_size), weights=fitness_scores, k=population_size)
   new_population = []

   for i in range(0, population_size, 2):
      parent1, parent2 = population[selected_indices[i]], population[selected_indices[i + 1]]
      crossover_point = random.randint(0, num_cities - 1)
      child1 = parent1[crossover_point:] + parent1[:crossover_point]
      child2 = parent2[crossover_point:] + parent2[:crossover_point]

      if random.random() < mutation_rate:
         swap_indices = random.sample(range(num_cities), 2)
```

```
            child1[swap_indices[0]], child1[swap_indices[1]] =
child1[swap_indices[1]], child1[swap_indices[0]]
        if random.random() < mutation_rate:
            swap_indices = random.sample(range(num_cities), 2)
            child2[swap_indices[0]], child2[swap_indices[1]] =
child2[swap_indices[1]], child2[swap_indices[0]]

        new_population.extend([child1, child2])

    population = new_population

best_tour = min(population, key=calculate_total_distance)
best_distance = calculate_total_distance(best_tour)

print("Best Tour:", best_tour)
print("Best Distance:", best_distance)
```