```
import random

# Function to create a random solution generator
def randomSolution(num_cities):
    cities = list(range(num_cities))
    solution = random.sample(cities, num_cities)
    return solution

# Function for calculating the length of a route
def routeLength(tsp, solution):
    route_length = 0
    num_cities = len(tsp)
    for i in range(num_cities):
        route_length += tsp[solution[i - 1]][solution[i]]
    return route_length

# Function for generating all neighbors of a solution
def getNeighbours(solution):
    neighbours = []
    num_cities = len(solution)
    for i in range(num_cities):
        for j in range(i + 1, num_cities):
            neighbour = solution[:]
            neighbour[i] = solution[j]
            neighbour[j] = solution[i]
            neighbours.append(neighbour)
    return neighbours

# Function for finding the best neighbor
def getBestNeighbour(tsp, neighbours):
    best_route_length = routeLength(tsp, neighbours[0])
    best_neighbour = neighbours[0]
    for neighbour in neighbours:
        current_route_length = routeLength(tsp, neighbour)
        if current_route_length < best_route_length:
            best_route_length = current_route_length
            best_neighbour = neighbour
    return best_neighbour, best_route_length

# Hill climbing algorithm
def hillClimbing(tsp, num_cities):
    current_solution = randomSolution(num_cities)
    current_route_length = routeLength(tsp, current_solution)
    neighbours = getNeighbours(current_solution)
    best_neighbour, best_neighbour_route_length =
getBestNeighbour(tsp, neighbours)

    while best_neighbour_route_length < current_route_length:
        current_solution = best_neighbour
        current_route_length = best_neighbour_route_length
```

```
        neighbours = getNeighbours(current_solution)
        best_neighbour, best_neighbour_route_length =
getBestNeighbour(tsp, neighbours)

    return current_solution, current_route_length

def main():
    num_cities = int(input("Enter the number of cities: "))
    tsp = []

    for i in range(num_cities):
        row = list(map(int, input(f"Enter the distances from city
{i+1} to all cities separated by spaces: ").split()))
        tsp.append(row)

    solution, route_length = hillClimbing(tsp, num_cities)

    print("Optimal Route:", solution)
    print("Optimal Route Length:", route_length)

if __name__ == "__main__":
    main()
```

```
Enter the number of cities: 5
Enter the distances from city 1 to all cities separated by spaces: 10
12 13 19 9
Enter the distances from city 2 to all cities separated by spaces: 12
13 10 7 9
Enter the distances from city 3 to all cities separated by spaces: 13
14 12 10 8
Enter the distances from city 4 to all cities separated by spaces: 12
13 13 10 7
Enter the distances from city 5 to all cities separated by spaces: 12
13 10 8 9
Optimal Route: [0, 1, 3, 4, 2]
Optimal Route Length: 49
```