153. You are given a list of cities represented by their coordinates. Develop a program that utilizes exhaustive search to solve the TSP. The program should:

1. Define a function distance(city1, city2) to calculate the distance between two cities (e.g., Euclidean distance).
2. Implement a function tsp(cities) that takes a list of cities as input and performs the following:
    - Generate all possible permutations of the cities (excluding the starting city) using itertools.permutations.
    - For each permutation (representing a potential route):
        - Calculate the total distance traveled by iterating through the path and summing the distances between consecutive cities.
        - Keep track of the shortest distance encountered and the corresponding path.
    - Return the minimum distance and the shortest path (including the starting city at the beginning and end).

Program:-

```
import itertools

import math


def distance(city1, city2):
    """ Calculate the Euclidean distance between two cities """
    x1, y1 = city1
    x2, y2 = city2
    return math.sqrt((x2 - x1)*2 + (y2 - y1)*2)


def tsp(cities):
    """ Solve the TSP using exhaustive search """
    n = len(cities)
    if n < 2:
        return 0, []

    # Generate all permutations of cities (excluding starting city)
    perm = itertools.permutations(cities[1:])

    min_distance = float('inf')
```

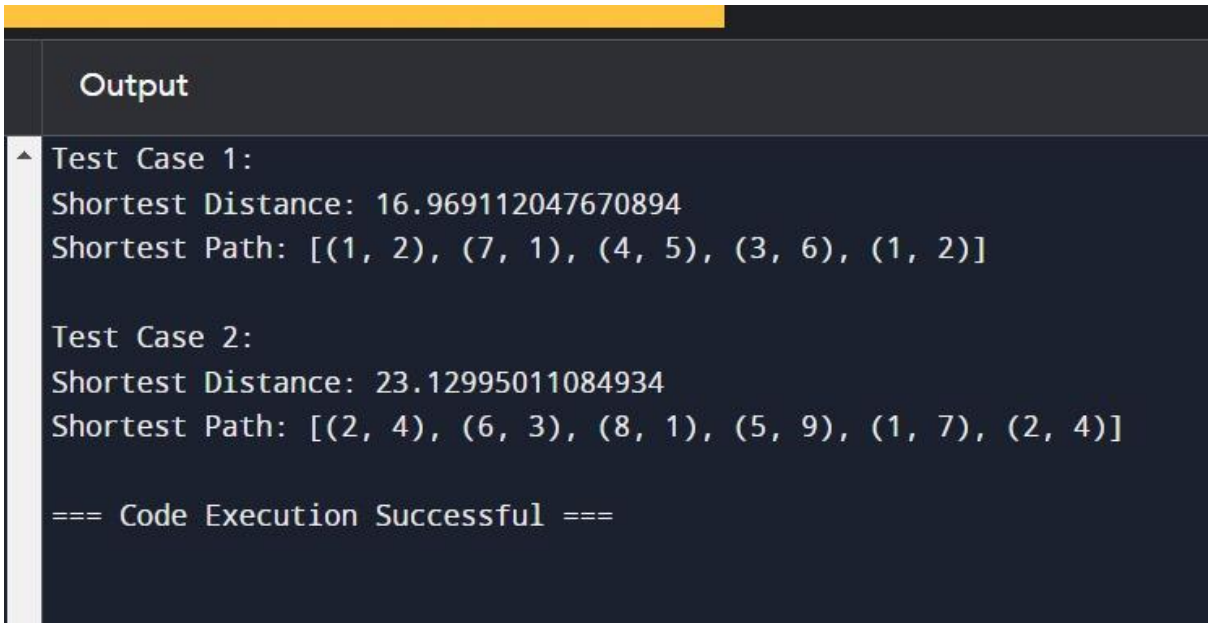
```
    shortest_path = None

    for route in perm:
        # Construct the full path including starting and ending at the first city
        full_route = [cities[0]] + list(route) + [cities[0]]

        # Calculate total distance for this route
        total_distance = 0
        for i in range(n):
            total_distance += distance(full_route[i], full_route[i+1])
  if total_distance < min_distance:
            min_distance = total_distance
            shortest_path = full_route

    return min_distance, shortest_path
```

input:-

cities1 = [(1, 2), (4, 5), (7, 1), (3, 6)]

cities2 = [(2, 4), (8, 1), (1, 7), (6, 3), (5, 9)]

output:-



TIME COMPLEXITY:-O(n-1)!