Testing Specification

1. **R1**
    a) Identified factors:
        i. Total distance of the calculated delivery path
            1. D1: Total path distance is well below the maxMoves field of the drone being used
            2. D2: Total path distance is approximately equal to the maxMoves field of the drone being used
            3. D3: total path distance is greater than the maxMoves field of the drone/drones being used, but is splitable across multiple trips/drones
            4. D4: Total path distance is a little bit greater than the maxMoves field of any drone and is not splitable.
            5. D5: Total path distance is much greater than the maxMoves field of any drone and is not splitable.
        ii. Number of orders
            1. N1: Single order
            2. N2: multiple orders
    b) Consider the following test cases:
        i. P1 - Simple case (D1 + N1)
            1. Input
                a) queries: A list of length one. We only have one order to deliver
                    i. The order is at a place where it is about 200 moves from the single service point we have
                b) drones: A list of length one, indicating we only have one drone.
                    i. maxMoves of the drone is set to 1000
                c) servicePoints: A list of length one, only one available drone base.
                d) servicePointDrones: A list of length one, corresponding to the one service point defined.
                    i. The only element of this list should indicate that the single drone we have is available at any time on any date.
                e) restrictedArea: An empty list, indicating no no-fly zones.
            2. Output
                a) True/False determined by an assert statement
                    i. True when the totalMoves field of the returned JSON object is less than or equal to the maxMoves attribute of the drone being used.
                    ii. False when the totalMoves field of the returned JSON object is greater than the maxMoves attribute of the drone

being used.

3. Specification
    a) It is guaranteed that the delivery can be made.
    b) The totalMoves attribute of the returned JSON object should be around 400.

ii. P2 - Boundary case (D2 + N1)

1. Input
    a) queries: A list of length one. We only have one order to deliver
        i. The order is at a location about 499 moves from the single service point we have.
        ii. 1 moves are reserved for hovering.
    b) drones: A list of length one, indicating we only have one drone.
        i. The drone's maxMoves is set to 1000.
    c) servicePoints: A list of length one, only one available drone base.
    d) servicePointDrones: A list of length one, corresponding to the one service point defined.
        i. The only element of this list should indicate that the single drone we have is available at any time on any date.
    e) restrictedArea: An empty list, indicating no no-fly zones.
2. Output
    a) True/False determined by an assert statement
        i. True when the totalMoves field of the returned JSON object is equal to the maxMoves attribute of the drone minus one.
        ii. False in all other cases
3. Specification
    a) It is guaranteed that the delivery can be made.
    b) The totalMoves attribute of the returned JSON object should be 999.
        i. 499 * 2 for delivering and returning, 1 for hovering, which gives us 999 moves in total

iii. P3 - Complex case (D3 + N2)

1. Input
    a) queries: A list of length two. We have two orders to deliver
        i. One order is placed 400 moves the North of the service

point

ii. The other order is placed 400 moves towards the East of the service point

iii. A singular tour visiting both orders will require more than 1000 moves

b) drones: A list of length one, indicating we only have one drone.

i. maxMoves of the drone is set to 1000

c) servicePoints: A list of length one, only one available drone base.

d) servicePointDrones: A list of length one, corresponding to the one service point defined.

i. The only element of this list should indicate that the single drone we have is available at any time on any date.

e) restrictedArea: An empty list, indicating no no-fly zones.

2. Output

a) True/False determined by an assert statement

i. True when the total moves of each delivery (service point -> delivery point -> service point) is less than or equal to the maxMoves attribute of the drone

ii. False when any of the deliveries has a move greater than the maxMoves attribute of the drone

3. Specification

a) It is guaranteed that the delivery can be made by making two separate deliveries.

b) The total moves of each delivery should equal to 801.

i. 400 * 2 for delivering and returning, 1 for hovering which gives us 801 moves in total.

iv. P4 – Edge Impossible Case (D4 + N1)

1. Input

a) queries: A list of length one. We only have one order to deliver

i. The order is at a place where it is 500 moves from the single service point we have.

b) drones: A list of length one, indicating we only have one drone.

i. maxMoves of the drone is set to 1000

c) servicePoints: A list of length one, only one available drone base.

d) servicePointDrones: A list of length one, corresponding to the one service point defined.

i. The only element of this list should indicate that the single drone we have is available at any time on any date.

e) restrictedArea: An empty list, indicating no no-fly zones.

2. Output
   a) True/False determined by an assert statement
      i. True when the totalMoves field of the returned JSON object is equal to 0, it should be impossible to find a path without exceeding the maxMoves attribute of the drone.
      ii. False when the totalMoves field of the returned JSON object is any values other than zero.
3. Specification
   a) It is guaranteed that the delivery cannot be made
   b) The totalMoves attribute of the returned JSON object should be zero.
      i. 500 * 2 for delivering and returning, 1 for hovering which gives us 1001 moves in total.
      ii. But 1001 > 1000, it is impossible to use this route therefore the algorithm should set the totalMoves to zero.

v. P5 - Impossible Case (D5 + N1)

1. Input
   a) queries: A list of length one. We only have one order to deliver
      i. The order is at a place where it is about 800 moves from the single service point we have
   b) drones: A list of length one, indicating we only have one drone.
      i. maxMoves of the drone is set to 1000
   c) servicePoints: A list of length one, only one available drone base.
   d) servicePointDrones: A list of length one, corresponding to the one service point defined.
      i. The only element of this list should indicate that the single drone we have is available at any time on any date.
   e) restrictedArea: An empty list, indicating no no-fly zones.
2. Output
   a) True/False determined by an assert statement
      i. True when the totalMoves field of the returned JSON object is equal to 0, it should be impossible to find a path without exceeding the maxMoves attribute of the drone.

ii. False when the totalMoves field of the returned JSON object is any values other than zero.

3. Specification
   a) It is guaranteed that the delivery cannot be made
   b) The totalMoves attribute of the returned JSON object should be zero.
      i. 800 * 2 for delivering and returning, 1 for hovering which gives us 1601 moves in total.
      ii. But 1601 > 1000, it is impossible to use this route therefore the algorithm should set the totalMoves to zero.

2. R2

a) Identified factors:

i. Position of no-fly zones relative to the “straight-line” route between service points and delivery points

1. RA1: No no-fly zones.
2. RA2: No-fly zones exist, but do not intersect any shortest path
3. RA3: A Singular no-fly zone intersects the shortest path, but alternative safe paths exist.
4. RA4: Multiple no-fly zones intersect many potential paths, which makes the alternative paths complex
5. RA5: No-fly zones block all paths between the service points and at least one delivery point

b) Consider the following test cases:

i. P1 – No no-fly zones case (RA1)

1. Input:
   a) queries: A list of length one, we only have one order to be delivered
      i. It is guaranteed that the order can be delivered via the shortest path.
   b) drones: a list of length one, we have one drone at the single service point.
   c) servicePoints: A list of length one, only one available drone base.
   d) servicePointDrones: A list of length one, corresponding to the one service point defined.
      i. The only element of this list should indicate that the single drone we have is available at any time on any date.
   e) restrictedArea: An empty list, we don’t have any no-fly zones.
2. Output:
   a) True/False determined by two assert statements
      i. True when no points on any sub-paths in the returned JSON object is in any restricted area (edges included) and the totalMoves field of the returned JSON object is not zero.
      ii. False in all other cases
3. Specification:
   a) None

ii. P2 – Singular no-fly zone not blocking shortest path case (RA2)

1. Input:
    a) queries: A list of length one, we only have one order to be delivered
        i. It is guaranteed that the order can be delivered via the shortest path.
    b) drones: a list of length one, we have one drone at the single service point.
    c) servicePoints: A list of length one, only one available drone base.
    d) servicePointDrones: A list of length one, corresponding to the one service point defined.
        i. The only element of this list should indicate that the single drone we have is available at any time on any date.
    e) restrictedArea: A list of length one. We have on no-fly zone
        i. This no-fly zone is placed far from the straight line path between the delivery point and the service point
2. Output:
    a) True/False determined by two assert statements
        i. True when no points on any sub-paths in the returned JSON object is in any restricted area (edges included) and the totalMoves field of the returned JSON object is not zero.
        ii. False in all other cases
3. Specification:
    a) None.

iii. P3 – Singular no-fly zone blocking shortest path case (RA3)

1. Input:
    a) queries: A list of length one, we only have one order to be delivered
        i. It is guaranteed that the order can be delivered via the shortest path.
    b) drones: a list of length one, we have one drone at the single service point.
    c) servicePoints: A list of length one, only one available drone base.
    d) servicePointDrones: A list of length one, corresponding to the one service point defined.

i. The only element of this list should indicate that the single drone we have is available at any time on any date.

e) restrictedArea: A list of length one. We have one no-fly zone

i. This no-fly zone is placed on the shortest path between the delivery point and the service point.

2. Output:

a) True/False determined by two assert statements

i. True when no points on any sub-paths in the returned JSON object are in any restricted area (edges included), and the totalMoves field of the returned JSON object is not zero.

ii. False in all other cases

3. Specification:

a) The shortest path should not be taken.

iv. P4 – Multiple no-fly zones blocking many paths case (RA4)

1. Input:

a) queries: A list of length one, we only have one order to be delivered

i. It is guaranteed that the order can be delivered via the shortest path.

b) drones: a list of length one, we have one drone at the single service point.

c) servicePoints: A list of length one, only one available drone base.

d) servicePointDrones: A list of length one, corresponding to the one service point defined.

i. The only element of this list should indicate that the single drone we have is available at any time on any date.

e) restrictedArea: A list of length three, we have three no-fly zones

i. The first one is placed on the shortest path between the delivery point and the service point.

ii. The second one is placed below the shortest path, and it touches the first no-fly zone.

iii. The third one is placed higher than the shortest path but does not touch the other two no-fly zones.

2. Output:

a) True/False determined by two assert statements

i. True when no points on any sub-paths in the returned JSON object are in any restricted area (edges included), and the totalMoves field of the returned JSON object is not zero.
ii. False in all other cases

3.
4. Specification:
   a) The shortest path should not be taken.

v. P5 – Multiple no-fly zones blocking all paths case (RA5)
   1. Input:
      a) queries: A list of length one, we only have one order to be delivered
         i. It is guaranteed that the order cannot be delivered via the shortest path.
      b) drones: a list of length one, we have one drone at the single service point.
      c) servicePoints: A list of length one, only one available drone base.
      d) servicePointDrones: A list of length one, corresponding to the one service point defined.
         i. The only element of this list should indicate that the single drone we have is available at any time on any date.
      e) restrictedArea: A list of length four, we have four no-fly zones
         i. The first one is placed on the shortest path between the delivery point and the service point.
         ii. The second one is placed below the shortest path, and it touches the first no-fly zone.
         iii. The third one is placed higher than the shortest path and touches the first two no-fly zones.
         iv. The last one is placed on the left and touches the second and the third no-fly zones. Together, these no-fly zones form a closed rectangle around the service point
   2. Output:
      a) True/False determined by two assert statements
         i. True when no points on any sub-paths in the returned JSON object are in any restricted area (edges included), and the totalMoves field of the returned JSON object is zero.

ii. False in all other cases

3. Specification:
   a) None

3. **R3**
    a) Identified factors:
        i. Mapping status of a drone:
            1. M1: The mapping is valid. The drone ID appears in at least one element of the list servicePointDrones, and the corresponding servicePointId exists in the servicePoints list.
            2. M2: The mapping is not valid. The drone ID appears in at least one element of the list servicePointDrones, but the corresponding servicePointId does not exist in the servicePoints list.
            3. M3: The mapping is not valid. The drone ID does not appear in any element of the list servicePointDrones.
        ii. Input composition
            1. C1: All drones are M1
            2. C2: There is at least one drone that is M2.
            3. C3: There is at least one drone that is M3.
            4. C4: There is at least one invalid-mapped drone (M2 or M3) and at least one valid-mapped drone (M1)
    b) Consider the following test cases:
        i. P1 - All drones have valid mapping (C1)
            1. Input:
                a) queries: A list of length one, we only have one order to be delivered
                    i. It is guaranteed that the order can be delivered.
                b) drones: a list of length three, we have three drones at the single service point. (D1, D2, D3)
                c) servicePoints: A list of length one, only one available drone base. (SP1)
                d) servicePointDrones: A list of length one, corresponding to the one service point defined.
                    i. The element should have its servicePointId set to SP1.id
                    ii. Since SP1.id is in servicePoints, D1, D2 and D3 are M1
                e) restrictedArea: An empty list, we don't have any no-fly zones
            2. Output:
                a) True/False determined by one assert statement
                    i. True when the planner uses at least one valid-mapped drones
                        1. totalMoves should not equal zero
                        2. dronePaths should not be an empty list

ii. False in all other cases

3. Specification:
    a) None

ii. P2 – At least one drone is M2 (C2)

1. Input:
    a) queries: A list of length one, we only have one order to be delivered
        i. The geography is feasible (no no-fly zones), but no eligible drones exist due to invalid mapping, so the planner should return no plan.
    b) drones: a list of length one, we have one drone at the single service point. (D1)
    c) servicePoints: An empty list, we don't have any available service points
    d) servicePointDrones: A list of length one, corresponding to the one service point defined.
        i. The element should have its servicePointId set to 999 (random number)
        ii. Since 999 is not in servicePoints, D1 is M2
    e) restrictedArea: An empty list, we don't have any no-fly zones
2. Output:
    a) True/False determined by one assert statement
        i. True when the planner cannot find a valid delivery path due to no drones can be used
            1. totalMoves should equal zero
            2. dronePaths should be an empty list
        ii. False in all other cases
3. Specification:
    a) None

iii. P3 – At least one drone is M3 (C3)

1. Input:
    a) queries: A list of length one, we only have one order to be delivered
        i. The geography is feasible (no no-fly zones), but no eligible

drones exist due to invalid mapping, so the planner should return no plan

b) drones: a list of length one, we have one drone. (D1)
c) servicePoints: An empty list, we don't have any available service points
d) servicePointDrones: An empty list
    i. Since we have a drone (D1), but servicePointDrones is empty, indicating the mapping is not valid and D1 is M3
e) restrictedArea: An empty list, we don't have any no-fly zones

2. Output:
    a) True/False determined by one assert statement
        i. True when the planner cannot find a valid delivery path due to no drones can be used
            1. totalMoves should equal zero
            2. dronePaths should be an empty list
        ii. False in all other cases
3. Specification:
    a) None

iv. P4 – A mix of different mapping (C4)

1. Input:
    a) queries: A list of length one, we only have one order to be delivered
        i. It is guaranteed that the order can be delivered.
    b) drones: a list of length three, we have three drones (D1, D2, D3)
    c) servicePoints: A list of length one, only one available drone base. (SP1)
    d) servicePointDrones: A list of length two
        i. The first element should have its servicePointId set to SP1.id and contain the availability of D1.
        ii. The second element should have its servicePointId set to 999 (random number) and contain the availability of D2
        iii. Since the element that contains the availability of D2 has a servicePoint ID of 999 (not in the servicePoint list), D2 is M2
        iv. Since we have drone D3, but there are no elements in servicePointDrones that contain the availability of D3, D3 is M3

e) restrictedArea: An empty list, we don't have any no-fly zones

2. Output:
   a) True/False determined by one assert statement
      i. True when the planner can find a path only using D1.
         1. totalMoves should not equal zero
         2. dronePaths should not be an empty list
         3. The droneId for every delivery (except the path returning to the service point) should be D1.id
      ii. False in all other cases
3. Specification:
   a) None

4. **R4**

   a) There is no need to write new test cases for this requirement. Simply record and analyse the time delay between sending out the request and getting the response should do the job.