Matrix distanceData[][] is a two dimensional list in Python:

distanceData[2][6] is the same as distanceData[6][2]

Index:		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	PackageID
		4001 South 700 East	1060 Dalton Ave S	1330 2100 S	1488 4800 S	177 W Price Ave	195 W Oakland Ave	2010 W 500 S	2300 Parkway Blvd	233 Canyon Rd	2530 S 500 E	2600 Taylorsville Blvd	2835 Main St	300 State St	3060 Lester St	3148 S 1100 W	3365 S 900 W	3575 W Valley Central Station bus Lo	3595 Main St	380 W 2880 S	410 S State St	4300 S 1300 E	4580 S 2300 E	5025 State St	5100 South 2700 West	5383 South 900 East #104	600 E 900 South	6351 South 900 East	
4001 South 700 East	0	0.0																											
1060 Dalton Ave S	1	7.2	0.0																										
1330 2100 S	2	3.9	7.1	0.0				**8.6**																					13,39
1488 4800 S	3	11.0	6.4	9.2	0.0																								
177 W Price Ave	4	2.2	5.0	4.4	5.6	0.0																							19
195 W Oakland Ave	5	3.5	4.3	2.8	6.9	1.9	0.0																						
2010 W 500 S	6	10.9	1.6	**8.6**	8.6	7.9	6.3	0.0																					
2300 Parkway Blvd	7	8.6	2.8	6.3	4.0	5.1	4.3	4.0	0.0																				
233 Canyon Rd	8	7.6	4.8	5.3	11.1	7.5	4.5	4.2	7.7	0.0																			
2530 S 500 E	9	2.8	6.3	1.6	7.3	2.6	1.5	8.0	9.3	4.8	0.0																		
2600 Taylorsville Blvd	10	6.4	7.3	10.4	1.0	6.5	8.7	8.6	4.6	11.9	9.4	0.0																	
2835 Main St	11	3.2	5.3	3.0	6.4	1.5	0.8	6.9	4.8	4.7	1.1	7.3	0.0																
300 State St	12	7.6	4.8	5.3	11.1	7.5	4.5	4.2	7.7	0.6	5.1	12.0	4.7	0.0															
3060 Lester St	13	5.2	3.0	6.5	3.9	3.2	3.9	4.2	1.6	7.6	4.6	4.9	3.5	7.3	0.0														
3148 S 1100 W	14	4.4	4.6	5.6	4.3	2.4	3.0	8.0	3.3	7.8	3.7	5.2	2.6	7.8	1.3	0.0													
3365 S 900 W	15	3.7	4.5	5.8	4.4	2.7	3.8	5.8	3.4	6.6	4.0	5.4	2.9	6.6	1.5	0.6	0.0												
3575 W Valley Central Statio	16	7.6	7.4	5.7	7.2	1.4	5.7	7.2	3.1	7.2	6.7	8.1	6.3	7.2	4.0	6.4	5.6	0.0											
3595 Main St	17	2.0	6.0	4.1	5.3	0.5	1.9	7.7	5.1	5.9	2.3	6.2	1.2	5.9	3.2	2.4	1.6	7.1	0.0										20,21
380 W 2880 S	18	3.6	5.0	3.6	6.0	1.7	1.1	6.6	4.6	5.4	1.8	6.9	1.0	5.4	3.0	2.2	1.7	6.1	1.6	0.0									
410 S State St	19	6.5	4.8	4.3	10.6	6.5	3.5	3.2	6.7	1.0	4.1	11.5	3.7	1.0	6.9	6.8	6.4	7.2	4.9	4.4	0.0								
4300 S 1300 E	20	1.9	9.5	3.3	5.9	3.2	4.9	11.2	8.1	8.5	3.8	6.9	4.1	8.5	6.2	5.3	4.9	10.6	3.0	4.6	7.5	0.0							14
4580 S 2300 E	21	3.4	10.9	5.0	7.4	5.2	6.9	12.7	10.4	10.3	5.8	8.3	6.2	10.3	8.2	7.4	6.9	12.0	5.0	6.6	9.3	2.0	0.0						15,16,34
5025 State St	22	2.4	8.3	6.1	4.7	2.5	4.2	10.0	7.8	7.8	4.3	4.1	3.4	7.8	5.5	4.6	4.2	9.4	2.3	3.9	6.8	2.9	4.4	0.0					
5100 South 2700 West	23	6.4	6.9	9.7	0.6	6.0	9.0	8.2	4.2	11.5	7.8	0.4	6.9	11.5	4.4	4.8	5.6	7.5	5.5	6.5	11.4	6.4	7.9	4.5	0.0				
5383 South 900 East #104	24	2.4	10.0	6.1	6.4	4.2	5.9	11.7	9.5	9.5	4.8	4.9	5.2	9.5	7.2	6.3	5.9	11.1	4.0	5.6	8.5	2.8	3.4	1.7	5.4	0.0			
600 E 900 South	25	5.0	4.4	2.8	10.1	5.4	3.5	5.1	6.2	2.8	3.2	11.0	3.7	2.8	6.4	6.5	5.7	6.2	5.1	4.3	1.8	6.0	7.9	6.8	10.6	7.0	0.0		
6351 South 900 East	26	3.6	13.0	7.4	10.1	5.5	7.2	14.2	10.7	14.1	6.0	6.8	6.4	14.1	10.5	8.8	8.4	13.6	5.2	6.9	13.1	4.1	4.7	3.1	7.8	1.3	8.3	0.0	

addressData[] **distanceData[][]**

(make sure spelling of addressee are identical to those in PackageData)

2-Opt Algorithm Implementation: Ref: https://www.youtube.com/watch?v=8vbKIfpDPJI

- The idea of 2-opt is simple: starting from a random tour, exchange two edges so that, when they are swapped, they produce a shorter tour.

- 2-Opt is a local search tour improvement algorithm. It is both a simple and fast heuristic to approximate the best tour.
- It originates from the idea that tours with edges that cross over are not optimal
- 2-opt will select two edges and replace them by two other edges if it results in an improved tour by visiting each node only once
- Swapping will continue until no improvement possible
- 2-opt Algorithm is O(n^2) time-complexity

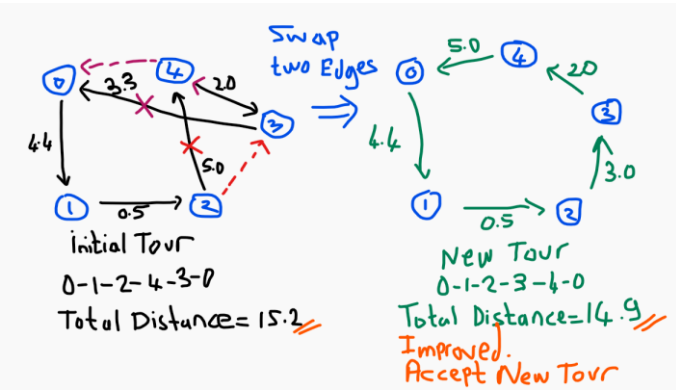
Pseudocode: Ref: https://en.wikipedia.org/wiki/2-opt

```
best_tour - Initial_tour (Random)
Repeat Until No Improvement {
   best_distance = calculateTotalDistance(best_tour)
   for (i = 0; i <= number of nodes eligible to be swapped - 1; i++) {
      for (j = i + 1; j <= number of nodes eligible to be swapped; j++) {
         new_tour = 2optSwap(best_tour, i, j)
         new_distance = calculateTotalDistance(new_tour)
         if (new_distance < best_distance) {
            best_tour = new_tour
            best_distance = new_distance
         }
      }
   }
}
```

For the DistanceMatrix above, if we assume the packages/addresses highlighted blue above are in Truck-1, we can create a DistanceMatrix specifically for Truck-1.
The ame can be done for all Trucks.
(this is just an assumption, trucks should be loaded based on the project requirements)

DistanceMatrix for Truck-1

Index:		0	1	2	3	4	PackageID
		1330 2100 S	177 W Price Ave	3595 Main St	4300 S 1300 E	4580 S 2300 E	
1330 2100 S	0	0.0					13,39
177 W Price Ave	1	4.4	0.0				19
3595 Main St	2	4.1	0.5	0.0			20,21
4300 S 1300 E	3	3.3	3.2	3.0	0.0		14
4580 S 2300 E	4	5.0	5.2	5.0	2.0	0.0	15,16,34

=>

0	1	2	3	4
0.0	4.4	4.1	3.3	5.0
4.4	0.0	0.5	3.2	5.2
4.1	0.5	0.0	3.0	5.0
3.3	3.2	3.0	0.0	2.0
5.0	5.2	5.0	2.0	0.0