Implement the UndergroundSystem class:

* void checkIn(int id, string stationName, int t)
  + A customer with a card id equal to id, gets in the station stationName at time t.
  + A customer can only be checked into one place at a time.
* void checkOut(int id, string stationName, int t)
  + A customer with a card id equal to id, gets out from the station stationName at time t.
* double getAverageTime(string startStation, string endStation)
  + Returns the average time to travel between the startStation and the endStation.
  + The average time is computed from all the previous traveling from startStation to endStation that happened **directly**.
  + Call to getAverageTime is always valid.

You can assume all calls to checkIn and checkOut methods are consistent. If a customer gets in at time **t1** at some station, they get out at time **t2** with **t2 > t1**. All events happen in chronological order.

**Example 1:**

**Input**

["UndergroundSystem","checkIn","checkIn","checkIn","checkOut","checkOut","checkOut","getAverageTime","getAverageTime","checkIn","getAverageTime","checkOut","getAverageTime"]

[[],[45,"Leyton",3],[32,"Paradise",8],[27,"Leyton",10],[45,"Waterloo",15],[27,"Waterloo",20],[32,"Cambridge",22],["Paradise","Cambridge"],["Leyton","Waterloo"],[10,"Leyton",24],["Leyton","Waterloo"],[10,"Waterloo",38],["Leyton","Waterloo"]]

**Output**

[null,null,null,null,null,null,null,14.00000,11.00000,null,11.00000,null,12.00000]

**Explanation**

UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(45, "Leyton", 3);

undergroundSystem.checkIn(32, "Paradise", 8);

undergroundSystem.checkIn(27, "Leyton", 10);

undergroundSystem.checkOut(45, "Waterloo", 15);

undergroundSystem.checkOut(27, "Waterloo", 20);

undergroundSystem.checkOut(32, "Cambridge", 22);

undergroundSystem.getAverageTime("Paradise", "Cambridge");       // return 14.00000. There was only one travel from "Paradise" (at time 8) to "Cambridge" (at time 22)

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 11.00000. There were two travels from "Leyton" to "Waterloo", a customer with id=45 from time=3 to time=15 and a customer with id=27 from time=10 to time=20. So the average time is ( (15-3) + (20-10) ) / 2 = 11.00000

undergroundSystem.checkIn(10, "Leyton", 24);

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 11.00000

undergroundSystem.checkOut(10, "Waterloo", 38);

undergroundSystem.getAverageTime("Leyton", "Waterloo");          // return 12.00000

**Example 2:**

**Input**

["UndergroundSystem","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime"]

[[],[10,"Leyton",3],[10,"Paradise",8],["Leyton","Paradise"],[5,"Leyton",10],[5,"Paradise",16],["Leyton","Paradise"],[2,"Leyton",21],[2,"Paradise",30],["Leyton","Paradise"]]

**Output**

[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]

**Explanation**

UndergroundSystem undergroundSystem = new UndergroundSystem();

undergroundSystem.checkIn(10, "Leyton", 3);

undergroundSystem.checkOut(10, "Paradise", 8);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.00000

undergroundSystem.checkIn(5, "Leyton", 10);

undergroundSystem.checkOut(5, "Paradise", 16);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.50000

undergroundSystem.checkIn(2, "Leyton", 21);

undergroundSystem.checkOut(2, "Paradise", 30);

undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 6.66667

**Constraints:**

* There will be at most 20000 operations.
* 1 <= id, t <= 106
* All strings consist of uppercase and lowercase English letters, and digits.
* 1 <= stationName.length <= 10
* Answers within 10-5 of the actual value will be accepted as correct.