Your application is a matching engine that receives orders and continuously matches them to see if they can trade. Your application should be implemented in Java or Kotlin.

The orders that your matching engine shall receive are time stamped upon arrival.
Orders are first ranked according to their price; orders of the same price are then ranked depending on the timestamp at entry.

Your application must read orders from stdin in the format described below. As your program reads the orders, it must emit any trades that result from the placement of an order. These trades shall be printed to stdout.

Once the input stream is fully consumed, your application must print all remaining orders that did not fully trade. These orders must also be printed to stdout. Please note the streaming nature of this input and structure your program accordingly.

Input format
Each order shall be delimited by a new line. Each line consists of an order's *OrderID*, *Side*, *Instrument*, *Quantity* and *Price*, delimited by a varying quantity of white space. The stream of orders are sequenced chronologically.

Sample Input
```
12345  BUY    BTCUSD   5   10000
zod42  SELL   BTCUSD   2   10001
13471  BUY    BTCUSD   6    9971
11431  BUY    ETHUSD   9    175
abe14  SELL   BTCUSD   7    9800
plu401 SELL   ETHUSD   5    170
45691  BUY    ETHUSD   3    180
```

Output format
For each trade produced by your application, your application must print a string followed by a newline. This string must begin with the word *TRADE* and be followed by the *Instrument*, *OrderID*, *ContraOrderID*, *Quantity*, and *Price* of the trade.

Once the input stream is fully consumed, your application shall print all orders that remain on each side of every order book. The *SELL* side of each order book shall be printed first; the *BUY* side of each order book 2 shall be printed last. For each side of every order book, orders shall be printed in the order in which they arrived. Note that the arrival order is different from each order's price/time priority.
Please ensure that you leave exactly one line of whitespace between each portion of the requested output.

Sample Output
```
TRADE     BTCUSD     abe14     12345      5      10000
TRADE     BTCUSD     abe14     13471      2       9971
TRADE     ETHUSD     plu401    11431      5       175

zod42          SELL          BTCUSD        2         10001
13471          BUY           BTCUSD        4          9971
11431          BUY           ETHUSD        4          175
45691          BUY           ETHUSD        3          180
```

Submission requirements:

Please submit an archived directory containing the following:
1. All source code and any additional tooling required to build, compile, and test your submission
2. A README.md containing the following:
    - instructions on how to build and run your application
    - a description of how you approached the problem
    - how long much time you spent on this project