Order Book Programming Exercise

Produce a program which maintains price-time limit order books, one per trading symbol. The program should accept new orders, order cancellations, and flushes from a CSV file and publish top of book (best bid and ask) changes for each order book. Supporting trades or matching is optional. See the details below.

Requirements

Input

Read a file of order transaction messages on an input thread. There are three types of transaction inputs:

- N = new order
- C = cancel order
- F = flush all orders

For the message format, see the provided input file `input_file.csv`.

Order Book Processing

An order book is price-time for bids and asks. An order joins its respective book side (i.e. bid or ask) in price then time priority.

Reject orders that cross the book.

Bonus (optional): Enable matching and trade orders that cross the book.

Output

Publish on output thread to the console/stdout. Use the output publishing formats below:

- New order or cancel order acknowledgement
  - **A, userId, userOrderId**
- Top of book change for a side, using '-' for `price` and `totalQuantity` where there are no orders on a side
  - **B, side (B or S), price, totalQuantity**
- Rejection for orders that would cross the book
  - **R, userId, userOrderId**
- Trade (matched orders) acknowledgement
  - **T, userIdBuy, userOrderIdBuy, userIdSell, userOrderIdSell, price, quantity**

Bonus: Create unit tests around the order book interface. As a shortcut, convert input scenarios to unit tests. As time permits, provide more scenarios.

Test Outputs

`output_file.csv` provides outputs for the scenarios in `input_file.csv`. Generate your own outputs for the scenarios. Validate your outputs against the outputs in `output_file.csv`, stripping the comments and blank lines out.

Project

Tar (or gzip) your project and e-mail it to the recruiter. Please do not include shared libraries, object files, or executables.

Provide a `README.md` file describing how to build and run the program.

Bonus: Document the project structure and any architectural aspects (i.e. threads, classes). Include improvements you would make if you had more time. Describe the time and space complexities of processing new orders and cancellations.

Bonus: Containerize the program and provide instructions with a Dockerfile to build and run via Docker.

Important Note: Please do not submit any code that is derived from proprietary code, including code you worked on for another company.