Order Book Challenge

You are asked to process some market data messages, build an order book (if you are not familiar with the idea of order book, please check out *Order Book* section below), and print output as the example shown in the *Output* section below.

There's a skeleton code at the end of this document. You can copy and paste it to your local code editor and start from there. We won't ask you to run your code. This challenge is more focus on how you would solve a practical problem.

We would recommend you going through the output example provided below carefully, which will give you an idea of what you are expected to achieve.

Market Data Specs

Message Type	Field Name	Data Type	Notes	Example
Add Order	msg_type	str	Message type, which is "A" for Add Order messages	"A"
	seq_num	int	Sequence number, starting from 1	99
	order_id	int	Unique order ID	287
	price	int	Order price	45
	quantity	int	Order quantity	20
	side	str	Order side, either "ask" or "bid"	"ask"
Modify Order	msg_type	str	Message type, which is "M" for Modify Order messages	"M"
	seq_num	int	Sequence number, starting from 1	99
	order_id	int	Unique order ID	287
	quantity	int	New order quantity	30
Delete Order	msg_type	str	Message type, which is "D" for Modify Order messages	"D"
	seq_num	int	Sequence number, starting from 1	99
	order_id	int	Unique order ID	287
Trade	msg_type	str	Message type, which is "T" for Trade messages	"T"
	seq_num	int	Sequence number, starting from 1	99
	order_id	int	Unique order ID	287

	counterparty_order_id	int	Order ID of the other side of the trade	120
	side	str	Side of the order, the ID of which is *order_id.*	“ask”
	price	int	Trade price	105
	quantity	int	Trade quantity	50

Order Book

Definition

An order book, essentially, is a list of current buy orders (also known as “bids”) and sell orders (also known as “asks”) for a specific asset. Order books show not only the price buyers and sellers are willing to pay, but also the total quantity they seek to buy or sell at each price.

A trade will occur when the book is crossed (best ask <= best bid and best ask > 0). In the example below, if a bid order at the price of 100 with a quantity of 50 is added to the book, the book will be crossed and a trade at the price of 100 with a traded quantity of 50 will be reported.

Example

Side	Price	Quantity	Note
Ask	…		
	110	300	This is the third level of ask price levels. The total quantity of ask orders with a price of 110 is 300.
	105	270	This is the second level of ask price levels. The total quantity of ask orders with a price of 105 is 270.
	100	150	**Best ask/Top level ask**. The total quantity of ask orders with a price of 100 is 150.
Bid	95	200	**Best bid/Top level bid**. The total quantity of bid orders with a price of 95 is 200.
	90	200	This is the second level of bid price levels. The total quantity of bid orders with a price of 90 is 200.
	85	330	This is the third level of bid price levels. The total quantity of bid orders with a price of 85 is 330.
	…		

Output

You are asked to print the top two levels' prices and quantities on both side of the book, as well as trade information. Here's an example of what the output looks like (please ignore the Event column, which is provided to help you understand how it works):

AP0	AQ0	AP1	AQ1	BP0	BQ0	BP1	BQ1	Message Type	SEQ_NUM	Trade Price	Trade Quantity	Event
100	40	105	15	95	10	90	5	A	10	NaN	0	**AddOrder(** seq_num: 10, msg_type: "A", order_id: 10, price: 100, quantity: 10, side: "ask" **)**
100	30	105	15	95	10	90	5	D	11	NaN	0	**DeleteOrder(** seq_num: 11, msg_type: "D", order_id: 10 **)**
100	50	105	15	95	10	90	5	A	12	NaN	0	**AddOrder(** seq_num: 12, msg_type: "A", order_id: 11, price: 100, quantity: 20, side: "ask" **)**
95	10	100	50	90	5	85	20	T	14	95	10	**AddOrder(** seq_num: 13, msg_type: "A", order_id: 12, price: 95, quantity: 20, side: "ask" **)** **Trade(** seq_num: 14, msg_type: "T", order_id: 12, counterparty_order_id: 8, side: "ask", price: 95, quantity: 10 **)**

Skeleton Code

```
import Reader


def main():
    reader = Reader()
    while reader.read("market_data.pcap"):
        print(reader.msg)   # print out current message
        # (msg is a Message object with attributes listed in the specs above)
    return 0


if __name__ == "__main__":
    main()
```