Simulation of Stock Trading

Problem definition

An online computer system for trading stock needs to process bids of the form “buy 100 shares at xx each” or “sell 50 shares at yy each.” A buy bid for xx can only be processed if there is an existing sell bid with price yy such that $y \leq xy \leq x$. Likewise, a sell bid for yy can only be processed if there is an existing buy bid with price xx such that $x \geq yx \geq y$.

If a buy or sell bid is entered but cannot be processed, it must wait for a future bid that allows it to be processed. Write a program that allows for buy and sell bids to be entered in O(log n) time, independent of whether or not they can be immediately processed.

The input file lists the bids in chronicle order, with the following fields:

```
bidId   clientId      action  price   shareCount
```

These fields are described next:

- bidId: ID of the bid, which is a zero or positive number in ascending order
- clientId: ID of the client, which ranges from 0 to 999.
- action: 0 for buy, 1 for sell, 2 for cancel
- price: Price to buy or sell (If action=2, then this field is actually bidId.)
- shareCount: No. of shares to buy or sell (If action=2, then this field is always 0.)

An typical example of input file is as follows:

```
0       983     1       114     2
1       741     0       128     41
2       816     0       183     68
3       244     1       171     100
4       712     1       125     1
5       543     1       127     61
6       934     1       144     74
7       234     0       173     68
```

```
8       48      1       181     58
9       883     0       139     39
.
.
.
```

The output lists all the possible transactions in chronicle order, with the following fields:

```
transId buyClientId     sellClientId    transPrice      transShareCount
```

For instance, the output corresponding to the previous example input is shown next:

```
0       741     983     114     2
1       816     244     171     68
2       741     712     125     1
3       741     543     127     38
4       234     543     127     23
5       234     934     144     45
.
.
.
```

Fields in both input and output are separated by tab.

Requirements & suggestions

- To generate the transaction list, you need to maintain a buy list and a sell list.
- To find the max. of the buy list, you need to implement it as a heap to support O(log n) tiime for "remove max".
- Similarly, to find the min. of the sell list, you need to implement it as a heap to support O(log n) time for "remove min".
- To make things easier, "cancel" can be done by linear scanning to have O(n) complexity. If you can implement "cancel" with O(log n) complexity, you'll be awarded with 10 extra points.
- You can safely assume that all the given data is well behaved without exceptions. (For instance, the price is always positive, etc.)
- The output is unique.

- If two same-price bids are available, the system should take the one with the smallest bidId first for any possible transactions. To ensure this, your heaps should be "stable" such that when there are two nodes with the same minimum in a min heap, the one with smaller bidId should be put at the root. (Same for max heaps.)
- TAs will use a test case to test the correctness of your program, with a time limit to determine if you use heaps or not.
- As usual...
    a. Your program should take the input from the standard input and send the output to the standard output.
    b. You need to write the program from scratch. You cannot use any open-source or readily available solvers.