Objective:

Create a matching engine that displays results based off of incoming orders.

Limit Order Stream (Input):

O/X/P | ID | SYM | B/S | Q | P

Result Stream (Output):

F/X/P/E | ID | SYM | B/S | Q | P

When do we result?

1. F – When a B & S order match by price, we match them by the minimum quantity
   1. If an order where B\_Price > S\_Price exists, then match them by S\_Price
2. X – When an order is canceled
3. P – When we want to print the current order book, by decreasing price
4. E – When ANY error occurs, be descriptive

What data structures will we need?

1. We need a data structure that will be able to hold the open orders, we should separate these by buy and sell limit orders
   1. We can use an unordered\_map for both buy and sell open limit orders, this will allow for O(1) lookup time when printing to results if we find a match. Key: price, Val: ID
   2. But this will only have a O(1) time complexity if we find exact price matches, what if we have an open order where B\_Price > S\_Price?
      1. Use MaxHeap for Buys
      2. Use MinHeap for Sells
      3. This comes at O(logN) time complexity for insertions, O(1) for top value
2. We need a data structure that will be able effectively print the open orders
   1. We can use a map<price, string{ID | SYM | B/S | Q}>, this will have O(logN) time complexity
3. For canceled orders, we simply print the canceled order ID
4. Errors will occur if:
   1. Input Syntax is incorrect
      1. Incorrect Order Action
      2. Duplicate Order ID, Non Positive 32-Bit Int
      3. Incorrect Symbol -> Either Empty string or greater than 8 size string
      4. Incorrect B/S Syntax (Anything other than B or S is input)
      5. Non Positive 16-Bit Int Quantity
      6. Non Positive Double for Price (int will be converted to 7.5 float)