OxFORD Asset Management
C++ Programming Test

This assignment is designed to test your ability to solve a simplified real-world problem in an efficient and straightforward manner.

We are interested in analysing a set of market data that consists of two types of message. Each time-stamped message is either an insert (of an "order" at a given price), or an erase (of an earlier order). You will need to write a class to maintain an "orderbook", which is the set of outstanding orders that have been inserted but not erased. It should be possible to ask for the highest price of the outstanding orders. You should then use this class in a console application which reads input data from a file and prints out the time-weighted average of the highest price.

The input data is in simple ASCII text format as follows:

```
1000 I 100 10.0
2000 I 101 13.0
2200 I 102 13.0
2400 E 101
2500 E 102
4000 E 100
```

Each line consists of either 3 or 4 fields, separated by spaces:

1. Timestamp (an integer, milliseconds since start of trading)
2. Operation (a single character, I = Insert, E = Erase)
3. Id (a 32-bit integer)
4. [Insert operations only] Price (a double-precision float)

In the data above, there are three time periods during which the high price is valid:

• 1000-2000	10.0
• 2000-2500	13.0
• 2500-4000	10.0

So the time-weighted average price is ((10 * 1000) + (13 * 500) + (10 * 1500)) / 3000 = 10.5

Note that there may be time periods in which there are no orders in the orderbook. These should not contribute to the output value.

You should implement your application in ISO standard C++98 or C++11, using standard libraries, ideally restricted to the STL and/or Boost. For simplicity the implementation should be provided as a single source code file.

The application should be a command-line tool which takes a filename and prints out the time-weighted average highest price to the console before exiting. The order book class should have a minimum of three methods: insert, erase and get highest price. The method to get the highest price should return NAN if the orderbook is empty.

You can assume:

- There is exactly one command-line argument (the filename)
- The file has been validated such that it is in the format above
- The timestamps are monotonically increasing
- Each order id appears exactly twice (one insert and one erase)
- Erase messages always appear after their corresponding insert
- The prices are always finite
- Calculation error due to double-precision arithmetic is not significant

Consideration will be given to:

- Correctness
- Efficiency
- Clarity, simplicity and elegance
- Effective re-use of standard libraries
- Focus on the problem in hand

You should not need to write a huge amount of code. A straightforward STL implementation is possible in under 100 lines of code.

Thank you for your interest.