# COMP226

## Assignment 1

**Rahul Savani**

rahul.savani@liverpool.ac.uk

# Reconstruct a Limit Order Book

| | |
|---|---|
| Continuous Assessment Number | 1 (of 2) |
| Weighting | 10% |
| Assignment Circulated | 09:00 Tuesday 18 February 2020 (updated 2020-02-20) |
| Deadline | 17:00 Friday 6 March 2020 |
| Submission Mode | Electronic only<br>http://www.csc.liv.ac.uk/cgi-bin/submit.pl<br>Submit a single file "MWS-username.R", where MWS-username should be replaced with your MWS username. |

| | |
|---|---|
| Learning Outcomes Assessed | Have an understanding of market microstructure and its impact on trading. |
| Goal of Assignment | Reconstruct a limit order book from order messages |
| Marking Criteria | Correctness (85%); Code Readability (15%) |
| Submission necessary in order to satisfy module requirements | No |
| Late Submission Penalty | Standard UoL policy; **resubmissions after the deadline will NOT be considered**. |
| Expected time taken | Roughly 8-12 hours |

## Learning Outcome

Have an **understanding of market microstructure** and its impact on trading

## Marking

- 85% **correctness** (determined primarily by automatic tests)

  **If your code passes all the tests you get the full 85%**

- 15% **readability** (5+5+5 for naming, comments, formatting)

## Code readability

- Use **good variable names** that describe what is being stored
    - `best_bid`, `executed_so_far`, rather than `x` and `y`

  & **good function names** that describe what the function does

- Write **informative comments**, e.g.,

```
# Check if the order id is in the book already
```

- Use **consistent spacing**

## Code/data zip handout

- **Download** `comp226_a1.zip`
- **Unzip** `comp226_a1.zip`
- This will yield a directory called `comp226_a1`

# Code/data zip handout

```
comp226_a1
├── input
│   ├── book_1.csv
│   ├── book_2.csv
│   ├── book_3.csv
│   ├── empty.txt
│   ├── message_a.txt
│   ├── message_ar.txt
│   ├── message_arc.txt
│   ├── message_ex_add.txt
│   ├── message_ex_cross.txt
│   ├── message_ex_reduce.txt
│   └── message_ex_same_price.txt
├── output
│   ├── book_1-message_a.out
│   ├── book_1-message_ar.out
│   ├── book_1-message_arc.out
│   ├── book_2-message_a.out
│   ├── book_2-message_ar.out
│   ├── book_2-message_arc.out
│   ├── book_3-message_a.out
│   ├── book_3-message_ar.out
│   └── book_3-message_arc.out
└── skeleton.R

2 directories, 21 files
```

## Code/data zip handout

- **skeleton.R**: **code template** that contains 6 empty functions that you need to complete
- **input**: subdirectory that contains two types of **input files**, initial book files and message files
- **output**: subdirectory that contains **sample output** that allows you to check your code implementations

## skeleton.R

```
$ Rscript skeleton.R input/book_1.csv input/empty.txt
$ask
  oid price size
1   a   105  100

$bid
  oid price size
1   b    95  100

Total volume:
Best prices:
Mid-price:
Spread:
```

`input/book_1.csv` is the initial book, `input/empty.txt` is the message file (empty in this case)

## "Main" part of code template

```r
if (!interactive()) {
    options(warn=-1)

    args <- commandArgs(trailingOnly = TRUE)

    if (length(args) != 2) {
        stop("Must provide two arguments: <path_to_book> <path_to_messages>")
    }
    book_path <- args[1]; data_path <- args[2]

    if (!file.exists(data_path) || !file.exists(book_path)) {
        stop("File does not exist at path provided.")
    }

    book <- book.load(book_path)
    book <- book.reconstruct(data.load(data_path), init=book)

    book.summarise(book)
}
```

# "Main" part of code template

- checks that there are two command line arguments
- assigns them to the appropriate variables (first to initial book file path, second to message file path)
- loads the initial book
- reconstructs the book according to the messages
- prints out the book
- prints out the book stats

## Rscript from Rstudio

```
Rscript skeleton.R input/book_1.csv input/empty.txt
```

- In R studio, you can call Rscript from the "terminal" tab (as opposed to the "console")
- On Windows, use Rscript.exe not Rscript:

```
Rscript.exe skeleton.R input/book_1.csv input/empty.txt
```

## 4 + 2 functions to implement

**Limit order book stats**:

1. `book.total_volume <- function(book)` **[10%]**
2. `book.best_prices <- function(book)` **[10%]**
3. `book.midprice <- function(book)` **[10%]**
4. `book.spread <- function(book)` **[10%]**

**Updating the limit order book**:

5. `book.reduce <- function(book, message)` **[15%]**
6. `book.add <- function(book, message)` **[30%]**

## input/book_1.csv

```
oid,side,price,size
a,S,105,100
b,B,95,100
```

| oid | side | price | size |
|---|---|---|---|
| a | S | 105 | 100 |
| b | B | 95 | 100 |

- `oid` (order id) is used to process (partial) cancellations of orders that arise in "reduce" messages
- `side`: 'B' for a buy/bid; 'S' for a sell/ask order
- `price` and `size` are self-explanatory

## Limit order book stats

- `book.total_volumes` should return **a list with named elements**, `bid` and `ask` where `bid` (`ask`) should be the total volume in the bid (ask) book
- `book.best_prices` should return **a list with two named elements**, `bid` and `ask` where `bid` (`ask`) should be the best bid (ask) price
- `book.midprice` should return the midprice
- `book.spread` should return the spread

## Expected output

```
$ Rscript solution.R input/book_1.csv input/empty.txt
$ask
  oid price size
1   a   105  100

$bid
  oid price size
1   b    95  100

Total volume: 100 100
Best prices: 95 105
Mid-price: 100
Spread: 10
```

## Message format

- message files contain one message per line (terminated by a single linefeed character, '`\n`')
- each message is a series of fields separated by spaces
- **two types of messages**: "Add" and "Reduce" messages.
- Here's an example, which contains an "Add" message followed by a "Reduce" message:

```
A c S 97 36
R a 50
```

## Add messages

```
'A' oid side price size
```

- '`A`': fixed string that identifies this as an "Add" message
- `oid`: "order id" used by subsequent "Reduce" messages
- `side`: '`B`' for a bid, '`S`' for an ask
- `price`: limit price of this order
- `size`: size of this order

## Reduce messages

```
'R' oid size
```

- 'R': fixed string identifying this as a "Reduce" message
- `oid`: "order id" identifies the order to be reduced
- `size`: amount by which to reduce the size of the order (*not* the new size of the order); if `size` is equal to or greater than the existing size of the order, the order is removed from the book

## Processing messages

- "Reduce" messages affect at most one existing limit order
- "Add" messages either:
    - **add a single row to the book** (orders at the same price are stored separately to preserve "oid"s)
    - **cross the spread** and then (partially) remove any number of orders on the other side of the book (and may result in a new limit order of unmatched volume)
- Example message files are split into cases that include crosses and those that don't to help you develop your code incrementally and test it on inputs of differing difficulty

## Ex: initial book

```
$ Rscript solution.R input/book_1.csv input/empty.txt
$ask
  oid price size
1   a   105  100

$bid
  oid price size
1   b    95  100

Total volume: 100 100
Best prices: 95 105
Mid-price: 100
Spread: 10
```

## Ex: processing a reduce message

```
$ cat input/message_ex_reduce.txt
R a 50
```

```
$ Rscript solution.R input/book_1.csv input/message_ex_reduce.txt
$ask
  oid price size
1   a   105   50

$bid
  oid price size
1   b    95  100

Total volume: 100 50
Best prices: 95 105
Mid-price: 100
Spread: 10
```

## Ex: Non-crossing add message

```
$ cat input/message_ex_add.txt
A c S 97 36
```

```
$ Rscript solution.R input/book_1.csv input/message_ex_add.txt
$ask
  oid price size
2   a   105  100
1   c    97   36

$bid
  oid price size
1   b    95  100

Total volume: 100 136
Best prices: 95 97
Mid-price: 96
Spread: 2
```

## Ex: crossing add message

```
$ cat input/message_ex_cross.txt
A c B 106 101
```

```
$ Rscript solution.R input/book_1.csv input/message_ex_cross.txt
$ask
[1] oid   price size
<0 rows> (or 0-length row.names)

$bid
  oid price size
1   c   106    1
2   b    95  100

Total volume: 101 0
Best prices: 106 NA
Mid-price: NA
Spread: NA
```

## 9 longer sample output files

| | messages_a | messages_ar | messages_arc |
|---|---|---|---|
| book_1 | | | |
| book_2 | | | |
| book_3 | | | |

```
output
├── book_1-message_a.out
├── book_1-message_ar.out
├── book_1-message_arc.out
├── book_2-message_a.out
├── book_2-message_ar.out
├── book_2-message_arc.out
├── book_3-message_a.out
├── book_3-message_ar.out
└── book_3-message_arc.out

0 directories, 9 files
```

## Hints for order book stats

For `book.spread` and `book.midprice` a nice implementation would use `book.best_prices`, which you should then implement first.

## Hints for `book.add` and `book.reduce`

A possible way to implement `book.add` and `book.reduce` that makes use of the different example message files is the following:

- First, do a partial implementation of `book.add`, namely implement add messages that do not cross. Check your implementation with `message_a.txt`.
- Next, implement `book.reduce` fully. Check your combined (partial) implementation of `book.add` and `book.reduce` with `message_ar.txt`.
- Finally, complete the implementation of `book.add` to deal with crosses. Check your implementation with `message_arc.txt`

## Submission

Remember to submit a single "MWS-username.R" file, where MWS-username should be replaced with your MWS username.

## ***Warning***

The department uses automatic plagiarism & collusion detection

**Do not discuss or show your work to other students**

Plagiarizers and colluders will likely receive a mark of zero

Several students were found to have plagiarised in previous years and this had serious consequences for their studies (**last year, two students had their studies terminated and left without a degree**).

## Measure of Software Similarity

http://theory.stanford.edu/~aiken/moss/

Moss is automatically run on **all pairs of submitted files**

It reports a **percentage similarity score**

**It can not be fooled by**:

- Reodering
- Change of variable/function names
- Different Comments

# Moss sample output

| File 1 | File 2 | Lines Matched |
|---|---|---|
| x1x1.R (99%) | x1x2.R (99%) | 110 |
| x1x4.R (99%) | x1x7.R (99%) | 110 |
| x1x3.R (99%) | x1z3.R (97%) | 24 |
| x1x3.R (51%) | x1x9.R (52%) | 46 |
| x1x3.R (48%) | x1x9.R (50%) | 44 |
| x1x6.R (53%) | x1z1.R (26%) | 26 |
| x1x9.R (28%) | x1x8.R (16%) | 26 |
| x1z2.R (29%) | x1z4.R (7%) | 15 |
| x1z3.R (15%) | x1x3.R (25%) | 23 |