Appendix B.I.b

To be provided to technical reviewers before interviewing Developer candidates

Notes for reviewers

Look out for the following things:

- what type do they use for currency? Do they understand the potential issues with this?
- do they continue to write tests before code?
- do they check the tests fail before they write code?
- are they able to describe their approach to you before jumping into the solution?
- have they considered relevant / obvious edge cases?
- give them some help naming things if they need it
- check they have well structured code
- if they're using Scala, is it functional?

Step 3: More complicated offers

- The shop adds Bananas which cost 20p
- Bananas are added to the same buy one, get one free offer as apples
- The cheapest item should be give free first
- Update your checkout

Step 4: Even more complicated offers

- The shop adds melons which cost £1
- Melons are available through a separate 3 for 2 offer
- Update your checkout

Step 5: Real time checkout

- Customers complain that they do not know how much their shopping costs until all produce are scanned
- Re-implement your checkout to show a running total as items are scanned.

Stretch Goal: Cheapest baskets

- Customers notice that they could save money by making multiple trips to the checkout. Why?
- Update the checkout so that customers get the lower price for only one trip

New content:

> *Tackle each step in order. Do not read ahead and do not try to anticipate specific future requirements. You are not expected to be able to finish this exercise*

Step 3: More complicated offers

- The shop adds Bananas which cost 20p
- Bananas are added to the same buy one, get one free offer as apples
- The cheapest item should be given free first
- Update your checkout

Step 4: Even more complicated offers

- The shop adds melons which cost £1
- Melons are available through a separate 3 for 2 offer
- Update your checkout

Step 5: Real time checkout

- Customers complain that they do not know how much their shopping costs until all produce are scanned
- Re-implement your checkout to show a running total as items are scanned.

Stretch Goal: Cheapest baskets

- Customers notice that they could save money by making multiple trips to the checkout. Why?
- Update the checkout so that customers get the lower price for only one trip

Pre-interview Exercise for Developer Candidates

The following exercise will assess your ability to deliver well structured and maintainable code. As a TDD shop we will put equal emphasis on evaluating your tests as we will on the code itself. The exercise should only take 30 - 40 minutes; don't spend much longer than this.

Please complete the following exercise in either Scala or Java. If this exercise goes well, you may be asked to make further amendments to your code during a pair programming exercise with a member of the team. Please pick the language you think you will be best able to demonstrate your skill and experience.

Please use git to version control your exercise. After completing the first step, please clearly tag the commit so that we can evaluate your approach. You may make other commits to further demonstrate your approach. Please compress your answers and submit them via your agent or send us a link to your Github (or similar) account. Do not send us binaries, they won't get through our firewalls.

> *Complete the steps in order. Don't read ahead. At each step build the simplest possible solution which meets our requirement. Tag a git commit after each step so that your approach is clear.*

Your answers will be used as part of our sifting and are likely to be discussed with your interviewer at later stages.

Step 1: Shopping cart

- You are building a checkout system for a shop which only sells apples and oranges.
- Apples cost 60p and oranges cost 25p.
- Build a checkout system which takes a list of items scanned at the till and outputs the total cost
- For example: [Apple, Apple, Orange, Apple] => £2.05
- Make reasonable assumptions about the inputs to your solution; for example, many candidates take a list of strings as input

Step 2: Simple offers

- The shop decides to introduce two new offers
    - buy one, get one free on Apples
    - 3 for the price of 2 on Oranges
- Update your checkout functions accordingly

In Person Exercises

The following exercises are to be used to assess candidates during in person interviews. Some exercises have notes for the candidate and separate notes for the interviewer.

> *Do not send this to the candidate or agent before the interview. It should be sent to the technical reviewers who will provide a copy during the in person interview.*

Appendix B.I.a

To be shown to Developer candidates at interview

Please complete the following exercise. At each stage build the simplest implementation which meets the requirement. Rather than looking stuff up, feel free to ask your partner for help.

Reminder of previously completed sections:

Step 1: Shopping cart

- You are building a checkout system for a shop which only sells apples and oranges.
- Apples cost 60p and oranges cost 25p
- Build a checkout system which takes a list of items scanned at the till and outputs the total cost
- For example: [Apple, Apple, Orange, Apple] => £2.05
- Make reasonable assumptions about the inputs to your solution; for example, many candidates take a list of strings as input

Step 2: Simple offers

- The shop decides to introduce two new offers
    - buy one, get one free on Apples
    - 3 for the price of 2 on Oranges
- Update your checkout functions accordingly