Exercise for Developer Candidates

The following exercise will assess your ability to deliver well-structured and maintainable code. The use of unit tests is encouraged. The exercise should only take 30 ­ 40 minutes; don’t spend much longer than this. Complete the steps in order.At each step build the simplest possible solution which meets our requirement.

We encourage the use of C# for this exercise as we are a .NET shop. However, you may 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 send us a link to your Github (or similar) account with the completed exercise.

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 45c and oranges cost 65c.
* Build a checkout system which takes a list of items scanned at the register and outputs the  total cost
* For example: [ Apple, Apple, Orange, Apple ] => $2.00
* 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

Additional Exercises

The following exercises are a continuation of the first exercise to evaluate your ability to create maintainable and unit testable code. We are again analyzing requirements and the ease of implementation and integration of code. There is not a specified time suggested for this section, but we are looking forward to reviewing as soon as possible.

* **Step 3: More complicated offers**
* The shop adds Bananas which cost 60c
* Bananas are added to the same buy one, get one free offer as apples
* The cheapest item should be given free (example: if you buy two apples and two bananas you get two apples for free)
* 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