REVIEW FEEDBACK

Jason Wong 17/04

17 April 2020 / 08:00 AM / Reviewer:

Steady - You credibly demonstrated this in the session.
Improving - You did not credibly demonstrate this yet.

GENERAL FEEDBACK

Feedback: Well done on a very good review, your process is now nice and steady across the course goals, which is great to see! Do remind me next time that you'd like to do one of these first three tasks again next time, so that you can do it in Javascript. Again, congratulations on your progress.

I CAN TDD ANYTHING – Strong

Feedback: I was impressed with your TDD process - you followed a nice and regular Red-Green-Refactor (RGR) cycle stuck to the excellent test progression you had laid out in your readme and didn't get ahead of yourself or the tests at any point. I was really pleased to see you identify that the core cases were a single correctly spelt and a single incorrectly spelt word, and that you addressed these right at the beginning with good tests based on the overall behaviour of the program. I was really impressed that you chose to implement a word bank, rather than using a gem from the word go - this decision is one of the 'tricky' bits of this review, and you 100% made the right decision. You really allowed the tests to drive the development of the code and so you spent very little time having to design the program or debug. This was a demonstration of the benefits of TDD - well done.

I CAN PROGRAM FLUENTLY – Strong

Feedback: It is clear that you are fluent with programming in Ruby. You were comfortable with the testing syntax, if-else statements, built-in methods like

include and split, as well as string interpolation and iteration with an each loop. You developed a really good algorithm to solve the problem and showed good insights into solving edge cases, notably the capitalisation issue.

I CAN DEBUG ANYTHING – Steady

Feedback:

I CAN MODEL ANYTHING – Steady

Feedback: You modelled your solution in a single method which I felt was a nice and simple implementation and provided a good place to start.

I CAN REFACTOR ANYTHING –Steady

Feedback: Refactors you discussed but didn't do:
After the first two tests you considered using a ternary to refactor the code, but you opted not to. This was a really good decision. A good rule of thumb not to refactor until you've moved beyond the core cases, as otherwise you risk removing or making it harder to see some very valuable structure that you have put in place that allowed you to differentiate between core cases.
You discussed switching the order of your if-else statements after the first two tests, to move the check to be for the correctly spelt word, rather than checking the incorrectly spelt word. This refactor would have been a valid one that doesn't violate the point I made above as it does not remove the structure, it simply makes it more expandable. I liked this thought and was sorry you didn't implement it as this would have been a baby step towards the next refactor.

Refactors you did do:
You refactored after your 4th test, which was an ideal opportunity to do so, developing a word bank to make the code more expandable so that it was possible to check against correctly spelt words instead of incorrectly spelt ones. This was a really great decision. A second refactor then generalised the return of the incorrectly spelt word. Also a great refactor.

I HAVE A METHODICAL APPROACH TO SOLVING PROBLEMS – Strong

Feedback: You have a lovely methodical approach to conducting a review in all aspects of your process. I would particularly like to note that your information gathering was very thorough with nice neat sections to guide you, and it was really nice to see that you checked off features that had been implemented in your readme. You also had good prioritisation of tasks, leaving edge cases till later. Your test progression was excellent and was structured in such a way as to provide value to the user in as fast a way as possible.

I USE AN AGILE DEVELOPMENT PROCESS – Strong

Feedback: You have developed a really nice and methodical information gathering process - well done! You asked some good questions about the input, establishing what the string could contain determining the extent of the task. You asked about the output and provided typed requirements which allowed me to clarify any discrepancies between our understandings of the requirements. You provided some typed examples, further checking we were on the same page and used these as a basis for your tests. You asked some good questions about edge cases, establishing the cases: empty string, non-string input, no multiple line strings, capitalisation and punctuation requirements.

I WRITE CODE THAT IS EASY TO CHANGE – Steady

Feedback: To save time you vocally let me know when you would commit to Git, and “committed” after almost every passing test, as well as after refactors. This was a good usage of Git.

I was pleased that you had your test suite properly decoupled from your implementation by making sure the tests were based solely on acceptance criteria, and not reliant on the current implementation. This makes changes to the code much easier.

I CAN JUSTIFY THE WAY I WORK – Strong

Feedback: The vocalisation of your process was really excellent. You let me know what test cases you were currently writing, you noted to me when you were moving onto the next phase of your RGR cycle (really nice, showed you were highly conscious of it), you let me know what errors you were expecting and how you what you were going to do to make tests pass. I would particularly like to mention that your thought process around the use of the word bank was excellent. This is a great skill to have, and will serve you well in technical interviews.