Real FizzBuzz

***** Please read the entire document before beginning. Pay close attention to the guidelines on page two***

Step 1:

Write some code that prints out the following for a contiguous range of numbers:

1. the number
2. 'fizz' for numbers that are multiples of 3
3. 'buzz' for numbers that are multiples of 5
4. 'fizzbuzz' for numbers that are multiples of 15

e.g. if I run the program over a range from 1-20 I should get the following output:

1. 1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16 17 fizz 19 buzz

Upload the source code to github as a separate project and provide the link to the repository

Step 2:

Enhance your existing FizzBuzz solution to perform the following:

1. If the number contains a three you must output the text 'lucky'. This overrides any existing behaviour

e.g. if I run the program over a range from 1-20 I should get the following output:

1. 1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz

Upload the source code to github as a separate project and provide the link to the repository

Step 3:

Enhance your existing solution to perform the following:

1. Produce a report at the end of the program showing how many times the following were output:
    a. Fizz
    b. Buzz
    c. Fizzbuzz
    d. Lucky

e. an integer

e.g. if I run the program over a range from 1-20 I should get the following output:

1. 1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz
   fizz: 4
   buzz: 3
   fizzbuzz: 1
   lucky: 2
   integer: 10
   (Integer is 10 because there were 10 numbers that were not altered in any way).

Upload the source code to github as a separate project and provide the link to the repository

Guidelines for offline code tests

1. You should not find this test to be particularly difficult. It is designed to be a straightforward coding exercise, and it should take you no more than 90 minutes
2. Please leverage PHP as the programming language and composer as your package manager. Your final result should work out of the box regardless of which environment is used (WAMP, LAMP, XAMPP, etc)
3. Please upload only your source files to github. Your application should be structured as follows:

   - yourapp/
   -- composer.json
   -- src/
   -- tests/

Things we are looking for

1. Test Coverage: The solution should be developed “test-first”, and should have excellent unit test coverage. If you are not familiar with unit testing in PHP, we advise you to use PHPUnit 8 (https://phpunit.de/getting-started/phpunit-8.html) and use PHP 7.2 with composer. It’s the easiest to set up.
2. Simplicity: We value simplicity as an architectural virtue and a development practice. Solutions should reflect the difficulty of the assigned task, and should not be overly complex. Layers of abstraction, patterns, or architectural features that aren’t called for should not be included.
3. Self-explanatory code: The solution you produce must speak for itself. Multiple paragraphs explaining the solution are a sign that it isn’t straightforward enough to understand purely by reading code, and are not appropriate.