Danphe Software Labs Ruby on Rails Engineer application screening test.

As part of the screening process, we expect an applicant for the post of Ruby/Rails engineer at Danphe Software Labs to complete the following tasks, before we sit down for an interview. The tasks - we think - are rather basic and help us to narrow down on serious and competitive applicants.

Ideally the solutions should be in Ruby - but you can choose any programming language of your preference for the solution, if you have a strong reason to do so.

Please create a github repo with the solutions to the problems, and email the link of the repo to us.

If you have any questions regarding the problems, please email us.

Good luck!

Problem 1:
Given three numbers X, Y & Z. write a function/method that finds the greatest among the numbers.

Problem 2:
Write a program that prints the number from 1 to 100. But for multiples of three print “Fizz” instead of the number & for the multiples of five print “Buzz”. However, for numbers which are multiples of both three and five print “ FizzBuzz” instead.

Problem 3:
Loop over a string of arbitrary length, and count the occurrence of each character in the string. Note: You can ignore accounting <whitespace>.
Eg:

Input: “hello how are you”
Output: h: 2, e: 2, l: 2, o: 3, : 3, w: 1, a: 1, r: 1, y: 1, u: 1

Problem 4:
Write a function/method in a **generic manner** such that it can convert from one number system to another; consider decimal to octal and binary. This method should take in three arguments as indicated below.

function convert_number(number, from, to)

Eg:
Input: *convert_number(42, “decimal”, “octal”)*

Output: 52

Input: *convert_number(42, “decimal”, “binary”)*
Output: 101010

Problem 5:
Write a method - let's call it *boxy(n)* - which produces output in STDOUT as shown below.

Example:

boxy(1)
Output:

boxy(3)
Output:

Problem 6:
Model the following entity relation requirement.

A system has many shops. A shop has many products, which can fall into one or many categories. The products can have different prices on different dates.

The solution to this can be an ER diagram with crow-foot notation, or it can be a text file that lists relevant classes and has active record relation statements.