Program 1: Ruby

This assignment consists of three parts and gives you an introduction to programming in Ruby. Please find the code template here. You need to complete the declared methods and class. **DO NOT** change the method or class definition in any way. Write your code where it says "`# ADD YOUR CODE HERE`".

Learning Goals

The objective of this assignment is that you will

- Understand the method signatures in Ruby
- Understand manipulating strings and arrays in Ruby
- Write simple code to use the basic object-oriented programming mechanisms in Ruby

Arrays, Hashes, and Enumerables

Check the official documentation on Array, Hash and Enumerable as they may be helpful in solving this exercise. Define the following methods:

1. **unique_total(a)** takes an array of integers as an argument and then returns the sum of unique elements in the array. The sum of an empty array is zero.
2. **highest_frequency(a)** which takes an array of integers as an argument and returns the element with highest frequency. If there is a tie, return the smallest element. For empty array, return -1.
3. **check_sum?(array, n)** that takes an array of integers and an additional integer, n, as arguments and returns true if any two elements in the array of integers sum to n. `checkSum?([], n)` should return false for any value of n, by definition.

Strings and Regular Expressions

Check the official Ruby documentation on String and Regexp as they may be helpful in solving this exercise. Define the following methods such that:

1. **custom_concat(s)** takes a string as an argument and returns the string "Welcome, " concatenated at the beginning of the input string.
2. **initial_consonant?(s)** takes a string as an argument and returns a boolean value. It is true if it starts with a consonant and false otherwise. (Any character other than a vowel is a consonant) This method should work for both uppercase and lowercase strings. Return false if string is empty or null.

3. **divisible_by_n?(s,n)** takes a string as an argument and returns a boolean value. It is true if the string represents a binary number that is exactly divisible by ‘n’ and false otherwise. It should return false even if the argument is not a valid binary number.

Object Oriented Basics

You will be implementing a class for `Car`. Each Car has two attributes - `model_number` and `price`. The attributes shouldn’t be publicly accessible; rather they should be read and modified through proper getters and setters.

The constructor should accept the `model_number` (a string) as the first argument and `price` as second argument, and should raise `ArgumentError` (one of Ruby's built-in exception types) if the `model_number` number is the empty string or if the `price` is less than or equal to zero.

Include a method `formatted_price` that returns the `price` of the `Car` formatted in the following manner:

- a `price` of `20000` should format as “20000 dollars only”
- a `price` of `20000.99` should format as “20000 dollars and 99 cents only”
- a `price` of `1.01` should format as “1 dollar and 1 cent only”
- a `price` of `0.60` should format as “60 cents only”

Testing

You should test all the methods of your code thoroughly. They should pass all general scenarios as well as edge cases. The test cases you use shouldn’t be part of assignment submission.

Scoring

You have to code 6 methods and 1 class. Each of the method carries 5 points and the class carries 5 points as well. Your method needs to pass all of our test cases to get full points.

Submission

Submit only the ruby_intro.rb file. You should rename the file as **ruby_intro_unityID1_unityID2.rb** before submitting.