CSC/ECE 517 (OO Design and Development)
Program 1: Ruby

Due Date: 09/13 Wednesday 11:59 PM

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". This file includes mock tests. You could run these tests and add more customized tests in order to check the correctness of your code. The only difference between mock tests and real tests we will run for grading is the number of test cases.

All YOUR FUNCTIONS SHOULD RETURN, NOT PRINT, THEIR RESULT VALUES.

Learning Goals

The objective of this assignment is that you will

- Understand 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. `two_sum?(a, n)` takes an array of integers `a` and an additional integer `n` as arguments and returns **true** if there are two different elements in `a` adding up to the integer `n`, and it returns false if not.
   Example:
   two_sum?([], 3) returns false
   two_sum?([3,4,5], 3) returns false
   two_sum?([3,1,2,2,4], 3) returns true

2. `max_sub_array(a)` Given an integer array `a`, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.
   A subarray is a contiguous part of an array.
   Example:

unique_array([1]) returns 1
unique_array([5,4,-1,7,8]) returns 23
unique_array([-2,1,-3,4,-1,2,1,-5,4]) returns 6

3. *group_anagrams(a)* Given an array of strings, group the anagrams together. You can return the answer in any order.
   An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.
   Example:
   group_anagrams([""]) returns [[""]]
   group_anagrams(["eat","tea","tan","ate","nat","bat"]) returns [["bat"],["nat","tan"],["ate","eat","tea"]]
   group_anagrams(["a"]) returns [["a"]]

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. *brackets_match?(s)* Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if:
   a. Open brackets must be closed by the same type of bracket.
   b. Open brackets must be closed in the correct order.
   Example:
   is_valid("()") returns true
   is_valid("()[]{}") returns true
   is_valid("(]") returns false
   is_valid("([)]") returns false
   is_valid("{[]}") returns true

2. *remove_and_append_vowels(s)* takes a string *s* representing a word as an argument and returns a string with all vowels of *s* removed and then appended to *s*. Characters are case-sensitive.
   Example:
   remove_and_append_vowels("Tree") returns "Tree"
   remove_and_append_vowels("run") returns "rnu"
   remove_and_append_vowels("Eyes") returns "ysEe"

3. highest_frequency_word(s) takes a string s that represents a sentence as an argument and returns the word (in lowercase) with highest frequency. If there is a tie, it returns the word that first appears earlier.
   Example: highest_frequency_word("Bob hit a ball; the hit BALL flew far after it was hit.") returns "hit"
   highest_frequency_word("A man was painting a new sign for the pub called the Pig and Whistle.") returns "a"
   highest_frequency_word("How are you doing?") returns "how"

Object-Oriented Basics

You will be implementing a Book class. Each book has two attributes, title 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 title (string) as the first argument and price as second argument, and should raise ArgumentError (one of Ruby's built-in exception types) if title is nil or an empty string, or if price is nil or less than or equal to zero.

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

- a price of 100 should format as "100 dollars only"
- a price of 10.49 should format as "10 dollars and 49 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 for all general scenarios as well as edge cases. The test cases you use shouldn't be part of assignment submission.

Scoring

You need to code 6 methods and 1 class. Each method is worth 5 points and the class is worth 5 points as well. Your method needs to pass all of our test cases to receive full credit.

Submission

Submit only the ruby_intro_fall23.rb file. You should rename the file as **ruby_intro_TeamName_unityID1_unityID2.rb** (or **ruby_intro_TeamName_unityID1.rb** if you are working alone) before submitting.
Example : ruby_intro_nah_jcui9_yxiao28

You can see your team name in Expertiza. All teams share a common submission area in Expertiza. Only one copy of the work should be submitted.