Assignment #2
CS 3060 Programming Languages, Fall 2020
Instructor: S. Roy

Ruby #2

Due Date: Sep 18. 11:59 pm

Total points: 60 points

Directions: Using the source provided via Gitlab `https://gitlab.com/sanroy/fa20-cs3060-hw/`, complete the assignment below. The process for completing this assignment should be as follows:

1. You already forked the Repository "sanroy/fa20-cs3060-hw" to a repository "yourId/fa20-cs3060-hw" under your username. If not, do it now.
2. Get a copy of hw2 folder in "sanroy/fa20-cs3060-hw" repository as a hw2 folder in your repository "yourId/fa20-cs3060-hw"
3. Complete the assignment, committing changes to git. Each task code should be in a separate ruby file. As an example, task1.rb for Task 1.
4. Push all commits to your Gitlab repository
5. If you have done yet done so, add TA and Roy as a member (in 'Developer' mode) of your Gitlab repository

Tasks:

1. **(7 points) Task #1:** Implement a program that takes a string x as input, and prints the lines of a file having any occurences of string x therein. The output should also show the line numbers of the matching lines. *Writing readme carries 1 point.*
   Example run: ruby task1.rb item-to-search (i.e. string x) file-to-search-in
   Note: 1. Your project repository needs to include a sample file-to-search-in. 2. Do NOT use any library's grep function. You implement it on your own, which may take only few lines of code.
2. **(8 points) Task #2:** Write a function that takes an array A of integers as the input and does the following: (a) Uses *each* method to print the square of each element of A, (b) Uses *each_slice* method to print the elements while printing 4 elements at a time, (c) Uses *select* method to find the integers (in A) which are multiple of 3, (d) Uses *map* method to build a new array of squares of the elements of A, and (e) Uses *inject* method to find the product of all elements of A. To test the function, generate an array A of 40 random integers between 10 and 80, and pass into the function as a parameter. *Writing readme carries 1 point.*
3. **(15 points) Task #3:**. Function3A calculates the Fibonacci Series ($F_n = F_{n-1} + F_{n-2}$) iteratively. Calculate all Fibonacci numbers (F_n) for $n = 1$ to 30. Function3B calculates the same Fibonacci Series but using the recursion technique. After implementing the above two functions, compare the computation time (for doing the above calculation) using Ruby's benchmark library, and also compare lines of code between the two implementations. *Writing readme carries 1 point.* **Hint**: if necessary, Ruby lecture slides (ppt) can help you write the iterative version.
4. **(15 points) Task #4:** The Tree class presented in the textbook (Day 2) chapter is interesting, but it does not allow you to specify a new tree with a clean user interface. Update the constructor and all other methods to accomodate the creation of a tree using a Hash. The initializer should accept a nested structure of Hashes. You should be able to specify a tree as below. To test your functionality, you should traverse this entire tree and print out the contents.

```
ruby_tree = Tree.new({
    'ggrandparent' => {
        'grandparent1' =>
            {'parent1' => {'child1' => {}},
```

```
                'parent2' => {'child2' => {}, 'child3' => {}}
            },
        'grandparent2' =>
            {'parent3' => {'child4' => {}},
             'parent4' => {'child5' => {}, 'child6' => {}}
            }
    }
})
```

Writing readme carries 1 point.

Hint: As one example coding style, you may mainly modify the *initialize* function of the Tree class.

5. **(15 points) Task #5:** In this task, your code creates a random list of 50 shape objects, then traverses the list from start to end, and computes the total area of the shape objects. First you need to implement the class hierarchy diagram of the shape types, which is attached. Shape is an abstract class which has only a "color" attribute whereas Circle class and Rectangle class are concrete children of Shape class, and they have more attributes and constructors. Note that you do not know beforehand the order of the shape objects (i.e. Circles and Rectangles) created in the random list, e.g. you do not know beforehand whether the 1st item is Circle or Rectangle. *Writing readme carries 1 point.*
   Note. When you traverse the list to calculate the total area, you call the area() function of each shape object (without considering if it is Rectangle or Circle). That means, you will use the concept of polymorphism.
   Hint: While building the list of shape objects, use rand(2) to generate a random number 0 or 1; if 0, then you may add a Rectangle object, else add a Circle object to the list.