**General instructions**

1. Read the question carefully, and note all the requirements that have been asked.

2. Look in your code, if there is any syntactic error.

3. Check all variable, method, file names and commit messages all should be meaningful.

4. Check Indentation and Spacing issue.

5. Check if there is any redundant code that you have written, try to optimize it.

6. Finally, strictly test all the requirements must be working in your program.

7. File names and method names should be snake\_case.

e.g email\_pattern.rb

def email\_pattern; end;

8. The class name should be CamelCase and would be the same as file\_name.

e.g. file\_name: credit\_card.rb, ClassName should be: CreditCard

9. Predicate methods must end with a ?

10. You should know about the methods you are using in your code. eg. collect, select, sort or any ruby method

11. Link your GitHub handler to your GitHub profile.

STEPS TO DO IT:

1. try: git config -l on your terminal.

2. If you find nothing, then try this:

# Set a new email

git config --global user.name "youremail"

e.g. git config --global user.name "test@example.com"

# Verify the setting

git config --global user.email # => test@example.com

12. Temporary files(ending with ~, .DS\_Store) should not be pushed on github. Also ensure that you do not push any unwanted file to repository.

13. Use {} for a single line block, and do end for multilines block.

14. Execute code and verify it is working as given in the question.

Exercise: 1 - Occurence Hash

Count the ocurrences of various alphabet letters in an input string and store it in hash. Your ruby program should accept a string as an argument and display the hash as an output.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"Hello World"

[/input]

[output]

{"H"=>1, "e"=>1, "l"=>3, "o"=>2, "W"=>1, "r"=>1, "d"=>1}

[/output]

===

[name]

3

[/name]

[input]

password123

[/input]

[output]

{"p"=>1, "a"=>1, "s"=>2, "w"=>1, "o"=>1, "r"=>1, "d"=>1}

[/output]

===

[name]

4

[/name]

[input]

@#goodbye\*%

[/input]

[output]

{"g"=>1, "o"=>2, "d"=>1, "b"=>1, "y"=>1, "e"=>1}

[/output]

#### Exercise: 2 - Replace using Regex Ask the user to enter text. Replace each vowel in the text with a '\*' using regular expression. Your program should accept a string as an argument and output the replaced string

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

password123

[/input]

[output]

p\*ssw\*rd123

[/output]

===

[name]

3

[/name]

[input]

"good morning"

[/input]

[output]

g\*\*d m\*rn\*ng

[/output]

===

[name]

4

[/name]

[input]

"hello everyone"

[/input]

[output]

h\*ll\* \*v\*ry\*n\*

[/output]

#### Exercise: 3 - Fibonacci using Yield

#### Write a program to print a Fibonacci series. Your program should accept a number input and output the Fibonacci series. Make use of yield in your program

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

1000

[/input]

[output]

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

[/output]

===

[name]

3

[/name]

[input]

150

[/input]

[output]

1 1 2 3 5 8 13 21 34 55 89 144

[/output]

===

[name]

4

[/name]

[input]

1

[/input]

[output]

1 1

[/output]

===

[name]

5

[/name]

[input]

2

[/input]

[output]

1 1 2

[/output]

===

[name]

6

[/name]

[input]

3

[/input]

[output]

1 1 2 3

[/output]

#### Exercise: 5 - Customer Account Balance

Define a class Account with three three attributes "name","account\_no" & "balance". Name and Balance should be set when creating an object of Account class, account\_number should be auto increment. This Account class must have two methods, deposit() and withdraw(). Your code should take three arguments  
  
customer one   
customer two   
transaction amount

Input Format:  
customer\_name1:account\_balance1 customer\_name2:account\_balance2 transfer:transaction\_amount

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

Rahul:2000 Abhishek:3000 transfer:200

[/input]

[output]

Account number: 1

Account holder name: Rahul

Account balance: 1800

Account number: 2

Account holder name: Abhishek

Account balance: 3200

[/output]

===

[name]

3

[/name]

[input]

"Shivam Goel":12500 "Jatin Kapoor":1000 transfer:12000

[/input]

[output]

Account number: 1

Account holder name: Shivam Goel

Account balance: 500

Account number: 2

Account holder name: Jatin Kapoor

Account balance: 13000

[/output]

Exercise: 6 Vehicle - Subclass  
Define a class named 'Vehicle' consisting of 'name','price' and methods for initializing and showing contents(overwrite to\_s). The 'price' of Vehicle may change over time. Now create a subclass 'Bike' having 'dealer' and percent\_price\_increase and method to show its content. Initialize a Bike class object with certain values. Define a method price\_increase which will increase the price. Your program should accept command line input in this format: BikeName Price Dealer PercentPriceIncrease. Ex: BajajDiscover 58000 BaggaLink 12

**Sample inputs**

Bike Name: BajajDiscover

Bike Price: 58000

Bike Dealer: BaggaLink

After 12.0 percent hike in price:

Bike Name: BajajDiscover

Bike Price: 64960.0

Bike Dealer: BaggaLink

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"Hero Honda" 65000 "Koncept Automobiles" 25

[/input]

[output]

Bike Name: Hero Honda

Bike Price: 65000

Bike Dealer: Koncept Automobiles

After 25.0 percent hike in price:

Bike Name: Hero Honda

Bike Price: 81250.0

Bike Dealer: Koncept Automobiles

[/output]

===

[name]

3

[/name]

[input]

apache 58000 "The Bike showroom" 12.5

[/input]

[output]

Bike Name: apache

Bike Price: 58000

Bike Dealer: The Bike showroom

After 12.5 percent hike in price:

Bike Name: apache

Bike Price: 65250.0

Bike Dealer: The Bike showroom

[/output]

#### Exercise: 4 - Palindrome Output whether the input string is a palindrome.Add a method palindrome? in String class, so that it can be called as str.palindrome? this method should return true/false. Input should be passed as a command line argument. \* When the string is Palindrome, the output should be "Input string is a palindrome" \* When the string is not Palindrome, the output should be "Input string is not a palindrome" \* If no argument is passed(or blank string argument). the output should be "Please provide an input" **Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

madam

[/input]

[output]

Input string is a palindrome

[/output]

===

[name]

3

[/name]

[input]

mamMAM

[/input]

[output]

Input string is not a palindrome

[/output]

#### Exercise: 7 - Inverse Case Overwrite the default 'to\_s' method such that it inverses the case of each letter. Eg: "hello WORLD".to\_s -> "HELLO world". Your program should accept a string as an argument and print the output on standard output.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

small

[/input]

[output]

SMALL

[/output]

===

[name]

3

[/name]

[input]

"G@@d MorninG"

[/input]

[output]

g@@D mORNINg

[/output]

#### Exercise: 8 Power - Array Define a method power() for an array. It takes an argument 'x' and returns the array with elements raised to power 'x'. Try to make use of array functions. Your program should accept an array and the value of power from command line. input example: ruby power\_array.rb "[1,2,3,4]" 2 output example: [1, 4, 9, 16] **Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"[1,2,3,4,5,6]" 3

[/input]

[output]

[1, 8, 27, 64, 125, 216]

[/output]

===

[name]

3

[/name]

[input]

"[-1,0,2,4,10]" 2

[/input]

[output]

[1, 0, 4, 16, 100]

[/output]

Exercise: 9 Array - Hash  
Create a method for Array that returns a hash having 'key' as the length of the element and value as an array of all the elements of that length. Make use of Array#each. Returned Hash should be sorted by key. Your program should accept command line arguments.   
  
Input argument: array-hash.rb "['abc','def',1234,234,'abcd','x','mnop',5,'zZzZ']"  
Output: {1=>["x", "5"], 3=>["abc", "def", "234"], 4=>["1234", "abcd", "mnop", "zZzZ"]}

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"['abc','def',1234,234,'abcd','x','mnop',5,'zZzZ']"

[/input]

[output]

{1=>["x", "5"], 3=>["abc", "def", "234"], 4=>["1234", "abcd", "mnop", "zZzZ"]}

[/output]

===

[name]

3

[/name]

[input]

"['hello','world',1,2,3,'good','bye']"

[/input]

[output]

{1=>["1", "2", "3"], 3=>["bye"], 4=>["good"], 5=>["hello", "world"]}

[/output]

Exercise: 10 Hash - Inject

Write a method that groups the above hash into 2 groups of 'even' and 'odd' length using 'inject'. Your ruby script should accept an array as command line argument and output the processed hash  
Eg: ['abc','def',1234,234,'abcd','x','mnop',5,'zZzZ'] -> {odd: [["x", 5], ["abc", "def", 234]], even: [[1234, "abcd", "mnop", "zZzZ"]]}  
  
**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"['abc','def',1234,234,'abcd','x','mnop',5,'zZzZ']"

[/input]

[output]

{"odd"=>[["abc", "def", "234"], ["x", "5"]], "even"=>[["1234", "abcd", "mnop", "zZzZ"]]}

[/output]

===

[name]

3

[/name]

[input]

"['hello','world',1,2,3,'good','bye']"

[/input]

[output]

{"odd"=>[["hello", "world"], ["1", "2", "3"], ["bye"]], "even"=>[["good"]]}

[/output]

===

[name]

4

[/name]

[input]

"['hello','world',1,2,3,'bye']"

[/input]

[output]

{"odd"=>[["hello", "world"], ["1", "2", "3"], ["bye"]]}

[/output]

#### Exercise: 11 Pascal - Yield

Print Pascal's triangle using 'yield'.

Eg: pascal(6) gives:

* 1
* 1 1
* 1 2 1
* 1 3 3 1
* 1 4 6 4 1
* 1 5 10 10 5 1

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

0

[/input]

[output]

[/output]

===

[name]

3

[/name]

[input]

2

[/input]

[output]

1

1 1

[/output]

===

[name]

4

[/name]

[input]

8

[/input]

[output]

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

[/output]

Exercise: 12 Character Count - Ranges  
Write a method that returns the no of various lowercase, uppercase, digits and special characters used in the string. Make use of Ranges.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"heLLo Every1"

[/input]

[output]

Lowercase characters = 7

Uppercase characters = 3

Numeric characters = 1

Special characters = 1

[/output]

===

[name]

3

[/name]

[input]

"g@@d MORNING"

[/input]

[output]

Lowercase characters = 2

Uppercase characters = 7

Numeric characters = 0

Special characters = 3

[/output]

===

[name]

4

[/name]

[input]

"my PASSw@rd i$ 123"

[/input]

[output]

Lowercase characters = 6

Uppercase characters = 4

Numeric characters = 3

Special characters = 5

[/output]

#### Exercise: 13 Factorial - Ranges

Rewrite the factorial method using ranges. Your script should take a number from the command line and output the result on standard output

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

0

[/input]

[output]

1

[/output]

===

[name]

3

[/name]

[input]

1

[/input]

[output]

1

[/output]

===

[name]

4

[/name]

[input]

5

[/input]

[output]

120

[/output]

===

[name]

5

[/name]

[input]

10

[/input]

[output]

3628800

[/output]

#### Exercise: 14 Reverse Sentence Use string methods to reverse the words arrangement in a sentence. Eg: "An apple a day keeps the doctor away" -> "away doctor the keeps day a apple An"

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"An apple a day keeps the doctor away"

[/input]

[output]

"away doctor the keeps day a apple An"

[/output]

===

[name]

3

[/name]

[input]

"away doctor the keeps day a apple An"

[/input]

[output]

"An apple a day keeps the doctor away"

[/output]

===

[name]

4

[/name]

[input]

"Hello everyone! Good morning"

[/input]

[output]

"morning Good everyone! Hello"

[/output]

#### Exercise: 15 Prime Numbers - Step Define a method to find all prime numbers upto n using 'step' function. **Don't use Prime class**

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

1

[/input]

[output]

[]

[/output]

===

[name]

3

[/name]

[input]

2

[/input]

[output]

[2]

[/output]

===

[name]

4

[/name]

[input]

20

[/input]

[output]

[2, 3, 5, 7, 11, 13, 17, 19]

[/output]

===

[name]

5

[/name]

[input]

100

[/input]

[output]

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

[/output]

Exercise: 16 Highlight Search Result  
Write a search method that searches for a letter or word in a user input string and highlights the searched element. Also it should display the total no of searches found. Eg: If user enters string to be searched as "Can you can a can as a canner can can a can?" and word to be searched is 'can' then output should be:   
(Can) you (can) a (can) as a (can)ner (can) (can) a (can)? Total Occurences found: 7

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"Can you can a can as a canner can can a can?" "can"

[/input]

[output]

(Can) you (can) a (can) as a (can)ner (can) (can) a (can)?

Total occurrences found: 7

[/output]

===

[name]

3

[/name]

[input]

"Can you can a can as a canner can can a can?" "can can"

[/input]

[output]

Can you can a can as a canner (can can) a can?

Total occurrences found: 1

[/output]

===

[name]

4

[/name]

[input]

"Good morning everyone" "hello"

[/input]

[output]

Good morning everyone

Total occurrences found: 0

[/output]

#### Exercise: 17 Sum Time

Write a method that sums up 2 24-hour time values(h:m:s). Validate the time using regular expressions. Eg: ("0:45:34","0:15:58") -> 01:01:32 ; ("11:23:07","22:53:45") -> 1 day & 10:16:52. **Note:** *Make use of Time class*

Exercise: 19 Sum Time  
  
Modify the sum of times method for any number of time values passed to this method. Eg: ("11:23:07","22:53:45","0:23:23","23:45:56") -> "2 day & 10:26:11"   
("11:23:07") -> "11:23:07"

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"24:01:10" "10:30:50"

[/input]

[output]

"Invalid 24-hour time value"

[/output]

===

[name]

3

[/name]

[input]

"0:45:34" "0:15:58"

[/input]

[output]

"01:01:32"

[/output]

===

[name]

4

[/name]

[input]

"12:45:2" "10:15:58"

[/input]

[output]

"23:01:00"

[/output]

===

[name]

5

[/name]

[input]

"11:23:07" "22:53:45"

[/input]

[output]

"1 day & 10:16:52"

[/output]

===

[name]

6

[/name]

[input]

"12:59:59" "12:1:1"

[/input]

[output]

"1 day & 01:01:00"

[/output]

#### Exercise: 20 Interest Difference Create a class Interest that gives the user the difference in amount calculated simply and compoundedly. Initialise class with a block. 'p' and 't' are entered by user. Take 'r' = 10% pa.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

10000 1

[/input]

[output]

Interest difference= 0.00

[/output]

===

[name]

3

[/name]

[input]

10000 2

[/input]

[output]

Interest difference= 100.00

[/output]

===

[name]

4

[/name]

[input]

15000 2.5

[/input]

[output]

Interest difference= 285.88

[/output]

Exercise: 21 Factorial  
Add exception handling for negative numbers in the factorial program.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

-5

[/input]

[output]

Negative number entered

[/output]

===

[name]

3

[/name]

[input]

0

[/input]

[output]

1

[/output]

===

[name]

4

[/name]

[input]

1

[/input]

[output]

1

[/output]

===

[name]

5

[/name]

[input]

5

[/input]

[output]

120

[/output]

Exercise: 22 Name - Raise

Create a class 'Name' with two attributes firstname and lastname. Neither of them can be blank and the first letter of firstname must be capital. Implement using 'raise'. Try using custom Exceptions.

**Sample inputs**

[name]

1

[/name]

[input]

manoj sharma

[/input]

[output]

Firstname must start with uppercase letter

[/output]

===

[name]

2

[/name]

[input]

Sachin

[/input]

[output]

Lastname cannot be blank

[/output]

===

[name]

3

[/name]

[input]

"Rakesh" "kumar"

[/input]

[output]

Your name is Rakesh kumar

[/output]

===

[name]

4

[/name]

[input]

Manoj Sharma

[/input]

[output]

Your name is Manoj Sharma

[/output]

#### Exercise: 23 CSV

* Read a CSV file:
* Name, EmpId, Designation
* Jack, 15, Developer
* Mary, 13, Designer
* John, 12, Developer
* Jane, 17, Designer
* Johny, 19, Tester
* Save into another file in the format:
* Designers
* Mary (EmpId: 13)
* Jane (EmpId: 17)
* Developers
* John (EmpId: 12)
* Jack (EmpId: 15)
* Tester
* Johny (EmpId: 19)

Listing should be in ascending order of Designation. Designation should be plural if it has more than one Employee.  
  
Exercise: Reverse Iterate  
  
Implement a reverse iteration function for Array such that I should be able to do this  
  
[2,4,6,8].reverse\_iterate { |i| print "#{i} "}  
It would print 8 6 4 2  
Note: Do not use any existing iterator for this.

**Sample inputs**

[name]

1

[/name]

[input]

[/input]

[output]

Please provide an input

[/output]

===

[name]

2

[/name]

[input]

"[2,4,6,8]"

[/input]

[output]

8 6 4 2

[/output]

===

[name]

3

[/name]

[input]

"[1.1,3.4,0,6,-23]"

[/input]

[output]

-23 6 0 3.4 1.1

[/output]

===

[name]

2

[/name]

[input]

"['hello', 01, 2, 3]"

[/input]

[output]

3 2 01 hello

[/output]

#### Exercise: Sales Tax Problem

Sales Tax Problem: You have to write a program in Ruby which would calculate the sales tax on the items according to the following conditions:

a. Flat 10 % sales tax is applicable on all items except for book, food and medicine  
b. Additional Import Duty of 5 % is applicable on all imported items without any exceptions

The input to the program would be in the following format:   
Name of the product: Chocolates  
Imported?: yes  
Exempted from sales tax? Yes  
Price: 120  
Do you want to add more items to your list(y/n): y  
Name of the product: Potato Chips  
Imported?: no  
Exempted from sales tax?: yes  
Price: 100  
Do you want to add more items to your list(y/n): y  
Name of the product: Perfume  
Imported?: yes  
Exempted from sales tax? No  
Price: 150  
Do you want to add more items to your list(y/n): n  
You have to generate a list that would display the list in a nice format and the grand total should be rounded to the nearest integer.