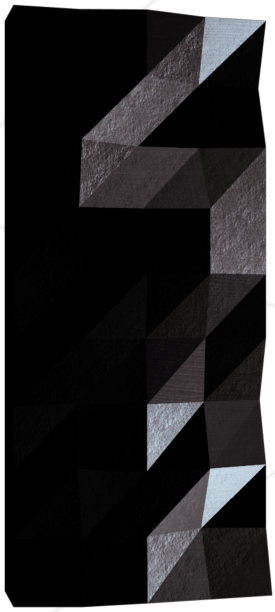

Atlantis D01

Dawn of the Second Day

Gaetan gaetan@42.us.org
Kai kai@42.us.org
Meo meo@42.us.org

Summary: Let us rejoice and let us sing and dance and ring in the new.

Contents

Chapter I

Guidelines

- Corrections will take place in the last hour of the day. Each person will correct another person according to the peer-corrections model.
- Questions? Ask the neighbor on your right. Next, ask the neighbor on your left.
- Read the examples carefully. The exercises might require things that are not specified in the subject...
- Your reference manual is called Google / "Read the Manual!" / the Internet / ...

Chapter II

Exercise 00 : ft_each

	Exercise 00
Array Traversal	
Turn-in directory : *ex*00/	
Files to turn in : `ft_each.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `ft_each.rb` which declares a number array [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and uses each method to iterate over the array and display each value.

```
$> ./ft_each.rb | cat -e
1$
2$
3$
4$
5$
6$
7$
8$
9$
10$
$>
```

`To each their own.`

Chapter III

Exercise 01 : Count on it

	Exercise 01
Counting and measuring parameters	
Turn-in directory : *ex*01/	
Files to turn in : `count_on_it.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `count_on_it.rb`, which, when executed, has "parameters:" and then the number of parameters, followed by a newline, then each parameter and its size followed by a newline. If there is no parameter, output `none` followed by a newline.

```
$> ./count_on_it.rb "Life" "Universe" "Everything" | cat -e
parameters: 3$
Life: 4$
Universe: 8$
Everything: 10$
$>
```

`Strictly adhere to the format indicated in the example.`

Chapter IV

Exercise 02 : Can you hear me now

	Exercise 02
Loops and loops and suddenly break!	
Turn-in directory : *ex*02/	
Files to turn in : `can_you_hear_me_now.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `can_you_hear_me_now.rb`. This script must contain a while loop that accepts a user input, write a return phrase, and stops only when the user has entered "YES!". Each round of loops must accept input from the user.

```
$> .can_you_hear_me_now.rb
Hello, is anyone there?: Hi
Can you hear me now?: Yes
Can you hear me now?: stop...
Can you hear me now?: YES!
Good!
$>
```

Chapter V

Exercise 03 : String Search

	Exercise 03
Getting to know strings	
Turn-in directory : *ex*03/	
Files to turn in : `string_search.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `string_search.rb` that takes a character string as a parameter. When executed, the script displays "z" for each character "z" in the string passed as a parameter, followed by a newline. If the number of parameters is different that 1, or there is no "z" character in the string, display "none" followed by a newline.

```
$> ./strings_search.rb "The target character is not in this string" | cat -e
none$
$> ./strings_search.rb "z" | cat -e
z$
$> ./strings_search.rb "Zealously zany zapping zebras zigzag zested Zoology zone" | cat -e
zzzzzzz$
$>
```

Chapter VI

Exercise 04 : ft_uniq_range

	Exercise 04
Array Manipulation	
Turn-in directory : *ex*04/	
Files to turn in : `ft_uniq_range.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `ft_uniq_range.rb` which takes an array of numbers (that you define) and builds a new array that is the result of adding the value 2 to each value of the original array. You must have two arrays at the end of the program, the original one and the new one you created. Display the two arrays on the screen using the `p` method rather than `puts`.
- You will only process values greater than 5 from the original array.
- No duplicates allowed!
  Note: you do not have to explicitly remove values from your arrays.
- For example, if your original array is [2, 8, 9, 48, 8, 22, -12, 2], you will get the following output:

```
$> ./ft_uniq_range.rb | cat -e
[2, 8, 9, 48, 8, 22, -12, 2]$
[10, 11, 50, 24]$
$>
```

`map`

Chapter VII

Exercise 05 : Pig Latin

	Exercise 05
Modifying strings	
Turn-in directory : $ex05/$	
Files to turn in : `atinlay.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script atinlay.rb that translates its arguments into pig latin.
- If the word starts with one or more consonants, remove the group of consonants and add them to the end of the string with a hypen. Then, add the letters "ay".
- If the first letter is a vowel, just add a "way" to the end.
- If there is no parameter, display none followed by a newline.
- You should probably downcase them, too

```
$> ./atinlay.rb "paranoid" "android" "Marvin" "so" "stoic" | cat -e
aranoidpay$
androidway$
arvinmay$
osay$
oicstay$
$>
```

Chapter VIII

Exercise 06 : ft_method

	Exercise 06
My first method	
Turn-in directory : $ex06/$	
Files to turn in : `ft_method.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `ft_method.rb` which includes a method. The method takes a string as an argument. The method must return an uppercase version of the string, if and only if the character string is longer than 10 characters. If the string is 10 characters or less, the method returns nil.

- Your program will call this method and display the return value on the command line. If there are no parameters, display none followed by a newline.

```
$> ./ft_method.rb | cat -e
none$
$> ./ft_method.rb "alo" | cat -e
nil$
$> ./ft_method.rb "hello world" | cat -e
nil$
$> ./ft_method.rb "i'M happY to be hERE" | cat -e
I'M HAPPY TO BE HERE$
$>
```

Chapter IX

Exercise 07 : do the math

	Exercise 07
Calculating averages	
Turn-in directory : $ex07/$	
Files to turn in : `do_the_math.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `do_the_math.rb` which contains a method `average_mark`. The method will use a hash, associating the first name of the students with their grade, to calculate the average score of the class on that test.

- So the following script:

```
$> cat do_the_math.rb
# your method definition here$

class_csci101 = {
"margot" => 17,
"june" => 8,
"colin" => 14,
"lewis" => 9
}
class_csci102 = {
"quentin" => 16,
"julie" => 15,
"mark" => 8,
"stephanie" => 13
}
puts "Average mark for the CSCI 101 class: #{average_mark class_csci101}."
puts "Average mark for the CSCI 102 class: #{average_mark class_csci102}."
$>
```

has the result:

```
$> /.do_the_math.rb | cat -e
Average mark for the CSCI 101 class: 12.$
Average mark for the CSCI 102 class: 13.$
```

Chapter X

Exercise 08 : ft_pedigree

	Exercise 08
Sorting hashes	
Turn-in directory : $ex08/$	
Files to turn in : `ft_pedigree.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Create a script `ft_pedigree.rb`. It will contain a `red_haired` method which takes in as a parameter a hash containing the first names of family members as key and their hair colors as attribute and return the names of all family members with red hair, sorted by alphabetical order.

- So the following script:

```
$> ./ft_pedigree.rb | cat -e
# your method definition here

Dupont_family = {
"matthew" => :red,
"mary" => :blonde,
"virginia" => :brown,
"gaetan" => :red,
"fred" => :red
}
p red_haired Dupont_family
$>
would have this result:
$> ./ft_pedigree.rb | cat -e
["fred","gaetan","matthew"]$
$>
```

Chapter XI

Challenge: save marvin

	Exercise 09
better known as, hangman	
Turn-in directory : $ex09/$	
Files to turn in : `save_marvin.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Code a version of Hangman named `save_marvin.rb` in the terminal.
- Display the secret word in underscores.
- If the user makes a correct guess, fill that in and display the partially completed word. It if is a failed guess, notify the user.
- Print a success message if the player guesses all letters in the word within X number of tries, determined by the user at the start of the game.

Chapter XII

Challenge: tic tac toe

	Exercise 10
the simplest tic-ing, tac-ing, and toe-ing	
Turn-in directory : $ex10/$	
Files to turn in : `tictac.rb`	
Allowed functions : `All`	
Notes : `n/a`	

- Build a working Tic Tac Toe program on the command line.
  We will display a game board, and take instructions from two players about where to put their pieces. The game board should show piece placement and be displayed after each move.
- You will turn in one single .rb file.
- It should print a welcome message to the terminal, `Welcome to Intergalactic Tic Tac Toe!` and the empty game board.
- To prevent cheating, insert a check to make sure that a players' move is valid.
- A move is not valid if it is outside the game board.
- A move is not valid if someone's x or o is already played there.
- Because we are nice, print out a congratulatory message for the winner at the end.

Chapter XIII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.