Ramda Functional Programming

COP 4530 Programming Project 6

Instructions

For Programming Project 6, you will be using Ramda (https://ramdajs.com) and Javascript to implement the functions below. Each of these functions should be made up of smaller more module functions that you create. I have given you some of the functions you will need as building blocks for the functions below, but you will need to make others to achieve the various tasks. Your functions will be operating on two JSON files, retrieved from the Flickr public API for images containing the tags dogs (dogs.json) and landscapes (landscapes.json).

You will need to use pure functional programming to create the functions below, but you should never be creating variables outside of arguments for functions and the functions themselves. Remember to use compose/pipe, Ramda functionality, and currying. If you do not use a functional approach to solve this problem, you will not receive any points.

Functions to Implement

`imageCount(flickrData)`
This function will return the number of images contained in the Flickr JSON data. The "items" property of the JSON holds a collection of objects that have the images data.

`alphaNumericTagsUniq(flickrData)`
This function will return an array of all the unique alphanumeric tags for all the images contained in the Flickr JSON data after transforming the tags to lower case. The tags returned by this method should first be checked to see if they are alphanumeric (A - Z upper/lowercase and 0 - 9), then made lowercase, then only the unique tags should be returned in the final array, sorted. The "items" property of the JSON holds a collection of objects that have the images data.

`nonAlphaNumericTags(flickrData)`
This function will return all non-alphanumeric tags (all letters other than A - Z upper/lowercase and 0 - 9) for all the images contained in the Flickr JSON data. This method does not sort, change case, or check for uniqueness. The "items" property of the JSON holds a collection of objects that have the images data.

`avgTitleLength(flickrData)`
This function will return a float with the average title length (in characters, including whitespace) of all the images contained in the Flickr JSON data. The "items" property of the JSON holds a collection of objects that have the images data.

`commonTagByRank(n)(flickrData)`

Turn the most common (frequently occurring) tag of rank n for all the images contained in the Flickr JSON data. Calling this functions with an n of 0 returns the most common tag, and calling it with an n of 2 returns the 3rd most common tag. The "items" property of the JSON holds a collection of objects that have the images data.

`oldestPhotoTitle(flickrData)`
This function will return the title of the oldest photo taken for all of the images contained in the Flickr JSON data. The “date_taken” property determines the age of the photo. The "items" property of the JSON holds a collection of objects that have the images data.

Examples

The testing section of the app.js file shows examples of the functions in use. The makePromiseTest function is purely used to test your code.

npm and Node.js

To work on this project, you will be using Node.js to run your Javascript code. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. You can install Node.js by downloading the package at https://nodejs.org/. After installing, you will have both Node and the Node Package Manager (npm). npm is a package manager for the JavaScript programming language and can be used to manage workspaces, install dependencies, and many other features.

After downloading the project files for this project, you will need to run the command ‘npm install’ from the directory with an active internet connection. The install command will install all dependencies of the project (like Ramda and others) so that you can start writing code. To run this project, you will type ‘node app.js’, which will run the file app.js that contains all of your code and the test code for this project.

Node.js and npm are powerful and useful tools that I highly suggest you read up on for your careers in and out of USF. Here is a playlist on YouTube with some videos about npm if you are interested. It is a little dated, but the messages are still useful: YouTube Playlist

Deliverables

Please submit complete projects as zipped folders. The zipped folder should contain:

- dogs.json (The Flickr JSON files you will operate on.)
- landscapes.json (The Flickr JSON files you will operate on.)
- package-lock.json (Used for npm. Do not alter.)
- testsetup.js (The code that sets up the tests. Do not alter.)
- app.js (Contains both your project code and the tests)
- extracredit.js (if you are submitting the extra credit)
- package.json (Used for npm. Do not alter.)

Make sure you remove the node_modules before submitting your code. npm installs this when the user runs ‘npm install’.

You should not need any additional files for this project.

Hints

Though Ramda may seem intimidating, it is a straightforward and powerful tool for functional/ declarative programming. The documentation for Ramda is extensive, useful, and necessary: https://ramdajs.com/docs/

If you are having trouble starting with Ramda, take a look at this cookbook (https://github.com/ramda/ramda/wiki/Cookbook), and for help on what functions do what, this is a helpful resource (https://github.com/ramda/ramda/wiki/What-Function-Should-I-Use%3F)

You can also google what other people have done using Ramda to get some inspiration.

You will want to spilt you code into smaller functions to make it more reusable and readable.

Utilize the functions I give you for free to make your work easier.

Remember how to use filter, map, and reduce because Ramda has its own version of these higher-order functions.

Use the R.comparator function when you write your functions with R.sort

Useful functions from Ramda for this project: R.map, R.sort, R.comparator, R.head, R.prop, R.nth, R.countBy, R.identity, R.toPairs, R.mean, R.uniq, R.filter, to just name a few. There are multiple solutions to these problems though.

If you are having trouble understanding JSON, this is a good resource: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/

Extra Credit

There is one opportunity for extra credit on this project. You will need to create a separate JS file (extracredit.js). When the file runs (using the command node extracredit.js), the small program will print out an image to the terminal from the dogs.json file. Below is an example.

The extra credit program takes one command line argument, that will be converted to a number between 0 and 19, representing which image from the dogs.json JSON file to print to the terminal. If an invalid number (outside 0 to 19 or just not a number) or has no value, you can assume the image to print is at image 0.

To get full credit for this extra credit (10 Points), you will need to use Ramda and JS to create functions that can access the nth link for an image using that command line argument (use the “m” property of “media” for the image link) from the Flickr data. Using that link, you will need to retrieve the image (I suggest using the “got” package) and then print the image to the terminal using the package “terminal-image” (https://github.com/sindresorhus/terminal-image).

You will need to mix pure and impure functional programming to achieve this goal, but you should never be creating variables outside of arguments for functions and the functions themselves. Remember to use compose/pipe, Ramda functionality, and currying. If you do not use a functional approach to solve this problem, you will not receive any points.

Rubric

Any code that does not run will receive a zero for this project.

Criteria	Points
flickrDataDog Tests (dogs.json)	
Images count should be 20	2
Should get an array of all the unique alphanumeric tags after transforming to lower case sorted lexicographically	2
Should Only Be 1 non alphanumeric tag as "świnoujście"	2
Average Title Length Should be 26 (Rounded)	3
Third most common tag should be "puppy"	3
Oldest Photo Taken Title Should Be "20160626_P1060675"	3
flickrDataLandscapes Tests (landscapes.json)	
Images count should be 20	2
Should get an array of all the unique alphanumeric tags after transforming to lower case sorted lexicographically	2
Should not be any non-alphanumeric tags	2
Average Title Length Should be 16 (Rounded)	3
Third most common tag should be "landscaping"	3
Oldest Photo Taken Title Should Be "Boats of Golyazi"	3
Other	
No variables (other than functions) are created and no state is mutated	6
Pipe or Compose Utilized for all functions	4
Total Points	**40**

Extra Credit

Criteria	Points
Prints a dog image out to the console	10
Total Points	**10**