CS320 - Homework #4

Due Nov 2th, 2020

Coding Standards (ESLint) and Git

1. Introduction

In this assignment, you will practice writing code in an environment where a code analysis tool is in place. Specifically, we will use ESLint, with our **.eslintrc** file, to enforce the AirBnB coding standards and use the provided **cs320-code-style.xml** file to enforce the layout of your code.

WARNING: if you haven't set these up yet, stop and go do the Code Inspection Tool Setup Guide first!

2. Problems to solve

The problems you are trying to solve in this assignment are as follows:

1) primeGen: This function takes a threshold number, and returns a list of prime numbers which are below that threshold.

E.g.,primeGen(10)=>[2, 3, 5, 7]

The Sieve of Eratosthenes would be useful here. **Do not have a hard coded list of prime numbers that gets used.** They should be generated with each call of a new value.

2) cumulativeSum: This function takes a list of numbers, and returns the cumulative sum of these numbers.

E.g.,cumulativeSum([1, 2, 3, 4])=>[1, 3, 6, 10]

For each value in the inserted array, it should be the sum of it's preceding values in the input array. So, position 2 is the sum of positions 0-2, which is 1+2+3 = 6 in the above example.

CS320 - Homework #4

Due Nov 2th, 2020

3) maxPrimeSum: This needs a bit more explanation:

The prime 41, can be written as the sum of six consecutive primes:

41 = 2 + 3 + 5 + 7 + 11 + 13

This is the longest sum of consecutive primes that adds to a prime below one-hundred. The longest sum of consecutive primes below one-thousand that adds to a prime, contains 21 terms, and is equal to 953.The function maxPrimeSum takes a threshold number, and returns a two-element array:

First, the prime number which is below the threshold and can be written as the sum of the most consecutive primes; and second the number of the consecutive prime terms.

```
maxPrimeSum(100)   => [ 41,  6]
maxPrimeSum(1000)  => [953, 21]
```

It would be a good idea to utilize the two previous functions for this part.

3. Workflow Instructions

You must follow the step-by-step workflow instructions below:

1. Create a private GitHub repo called "primefunctions" and **clone** it to your computer. You may need to apply for a "Student Developer Pack" in order to do this: https://education.github.com/pack

2. Create an IntelliJ Javascript project within your local "primefunctions" repo directory. If IntelliJ asks if you want to add a .gitignore file, you can say yes. Add "*.iml" and ".idea/" to the .gitignore file, so these IDE related files won't be tracked.

CS320 - Homework #4

Due Nov 2th, 2020

3. Set up your project to use our Javascript coding standards. See the “ESLint Configuration” guide, in the guide section of Blackboard, for details. In a nutshell it involves:

- Select the cs320-code-style.xml preferences template (File -> Settings -> Editor-> Code Style)
- Disable IntelliJ Javascript Inspections (File -> Settings -> Editor -> Inspections -> Javascript)
- Define the Javascript Language as ECMAScript6 (File -> Settings -> Languages & Frame-works -> Javascript)
- Enable ESLint (File -> Settings -> Languages & Frameworks -> Javascript -> Code Quality Tools -> ESLint)
- Download .eslintrc into the project directory.

4. Create two files: index.html and primefunctions.js. The index.html should load the Under-score package using
<script src="//cdnjs.cloudflare.com/ajax/libs/underscore.js/1.8.3/underscore-min.js"></script>

Then you should load "primefunctions.js" after that. Note that the order of loading is important! If you use any underscore functions, you should run your program in a browser with web developer tools accessible. Check the ESLint_demo code on Blackboard if you are confused about this step. Note that using underscore functions is **NOT** mandatory for this assignment. But in either case, you should create this index.html file as instructed.

(Optional) Once the set up is completed making an “initial commit” of your project is a good idea. Consider doing so now so you’ll have your starting point saved.

CS320 - Homework #4

Due Nov 2th, 2020

5. Write function primeGen, and make sure it works as expected.

6. Commit your project to GitHub, label it "Finished primeGen." Check that you code is on GitHub. It must be a git based service hosted on an external source.

7. Write function cumulativeSum, and make sure it works as expected.

8. Commit your project to GitHub, label it "Finished cumulativeSum." Check that you code is on GitHub.

9. Write function maxPrimeSum, and make sure it works as expected.

10.Make sure that there is a green check mark indicating that ESLint does not detect any problems in your code.

11. Commit your project to GitHub, label it "Finished maxPrimeSum." Check that you code is on GitHub. Now your GitHub should have a list of commits for the project, from start to finish.

The commit entry list should at least have:

Finished maxPrimeSum
Finished cumulativeSum
Finished primeGen
Initial commit (optional)

A picture of your Git commit list will be submitted for this assignment. Remember, you must be using a git based external service. A picture of your local repo commits will not suffice.

CS320 - Homework #4

Due Nov 2th, 2020

4. Submission

This assignment is due at 11:59pm on 11/2/2019. Late submissions are subject to a 20% penalty. Late submissions will not be accepted after 3 days past the deadline, in this case 11:59pm on 11/5.

Submit the following files in a compressed file (.zip or .tar) to Blackboard:

1. index.html

2. primefunctions.js

3. .gitignore

4. A GitHub (or equivalent git based external service site) screenshot of your assignment commits. This should have the 3 “finished” commits (one for each completed function), and potentially the optional “initial commit”.

5. Grading

This assignment will be graded out of 70. For your information, the grading scheme is showing the following table.

15pts primeGen
10pts cumulativeSum
20pts maxPrimeSum
10pts Use of Git
15pts Code quality (no ESLint erros)

TOTAL: 70 pts