Assignment 04 – Sum of Primes (loops)

THE PROBLEM: Write a program that computes the sum of the prime numbers between two inputs, X and Y. As in previous assignments: If a user provides any input that is not a natural number, then the function should return the error message: "Bad data. Try again."

What's a prime number? In mathematics, a prime number is any number that has exactly two positive divisors: 1 and itself.

For example: 7 is prime.

7 / 1 = 7 (a positive divisor!)
7 / 2 = 3.5
7 / 3 = 2.333...
7 / 4 = 1.75
7 / 5 = 1.4
7 / 6 = 1.1666...
7 / 7 = 1 (a positive divisor)

7 is prime because 7 has ***only two*** positive divisors: 1 and 7.

Another example: 8 is not prime.

8/1 = 8 (a positive divisor!)
8/2 = 4 (a positive divisor!)
8/3 = 2.666…
8/4 = 2 (a positive divisor)
8/5 = 1.6
8/6 = 1.333…
8/7 = 1.142857…
8/8 = 1 (a positive divisor)

8 is not prime because 8 has ***four*** positive divisors: 1,2,4, and 8.

How do you compute the sum of primes? Well… you see… you start counting at the beginning (for example, at 1) and then you don't stop counting until you reach the end (for example, 10). At each step along the way, you ask yourself, is this number prime? If it is, add it to your sum. If it's not, don't do anything and then move on to the next number.

The sum of primes between 1 and 10 would be 17 because:

2 + 3 + 5 + 7 = 17

We did not include the numbers 1,4,6,8 and 9 in our summation because those numbers aren't prime.

HOT TIP: You can reuse your isNaturalNumber() function from a prior assignment to help you with your error trapping.

ANOTHER HOT TIP: This assignment will require you to use the modulus operator (oh no! more math stuff!) To get an overview of this operator see: https://www.w3schools.com/js/js_arithmetic.asp

FUN FACT: Mathematicians tell us the 1 is not a prime number. Accept that as true. One is not prime.

ANOTHER FUN FACT: Prime numbers play an incredibly important role in Cryptography, Information Security and Crypto Currencies. So, while the purpose of this assignment is to get you writing some loops, it also introduces you to this idea: a computer must do quite a lot of work to generate a prime number sequence!

YET ANOTHER FUN FACT: In this assignment we will create an isPrime() function. While it is not presented here, a simple tweak to the isPrime() function could make it run roughly twice as fast. If you want to challenge yourself, you might want to think about how that could be done. It's more of a math/logic problem than a technical problem. This is totally optional.

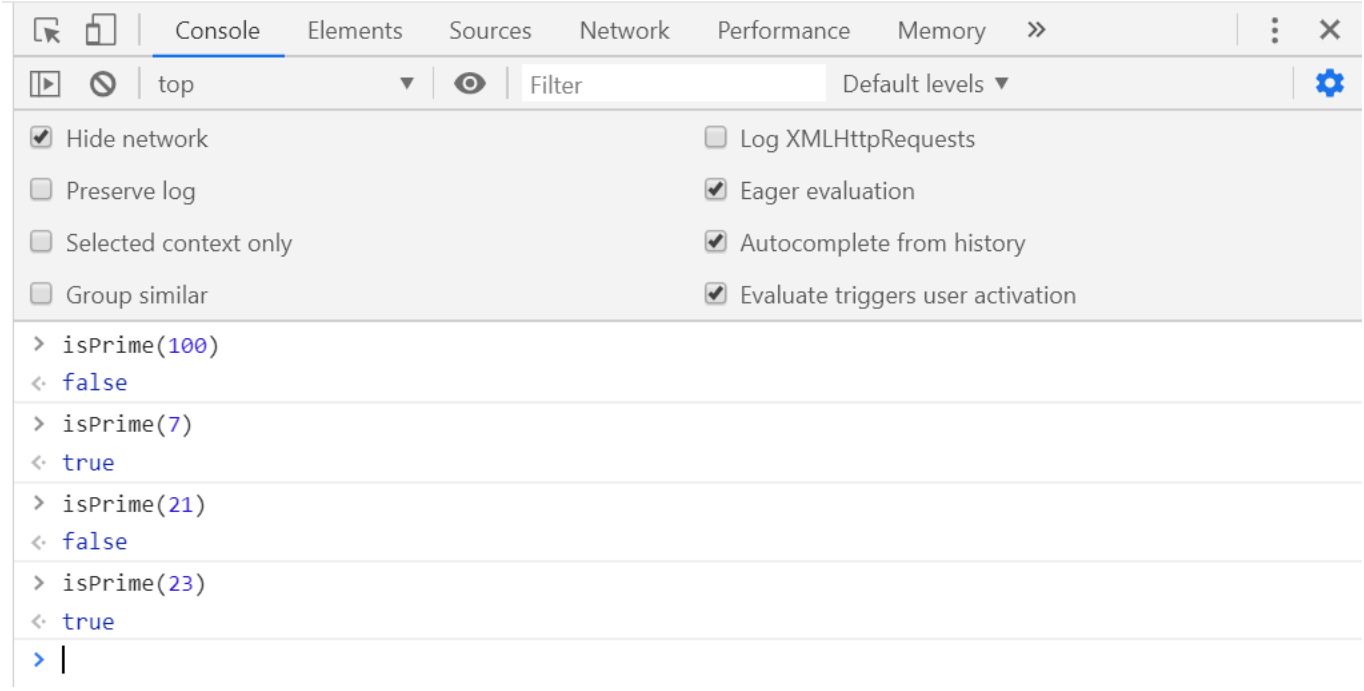
Instructions

1. Retrieve assignment04.zip provided by your instructor.
2. Extract the code into your mis2402workspace and open the primes.html file in Visual Studio Code.
3. Oh no! It looks like we have some sort of "Unexpected End of Input" error here. How do we solve that?
4. That wasn't the only error in the start file, can you find the other one?
5. Bring in the isNaturalNumber() function from a previous assignment. Put it in portion of your code designated for "supporting functions".
6. Use the web developer tools in Chrome to test your isNaturalNumber() function.
7. Complete the isPrime () function. The isPrime() function should return true if the provided parameter is prime. Otherwise the function should return false.

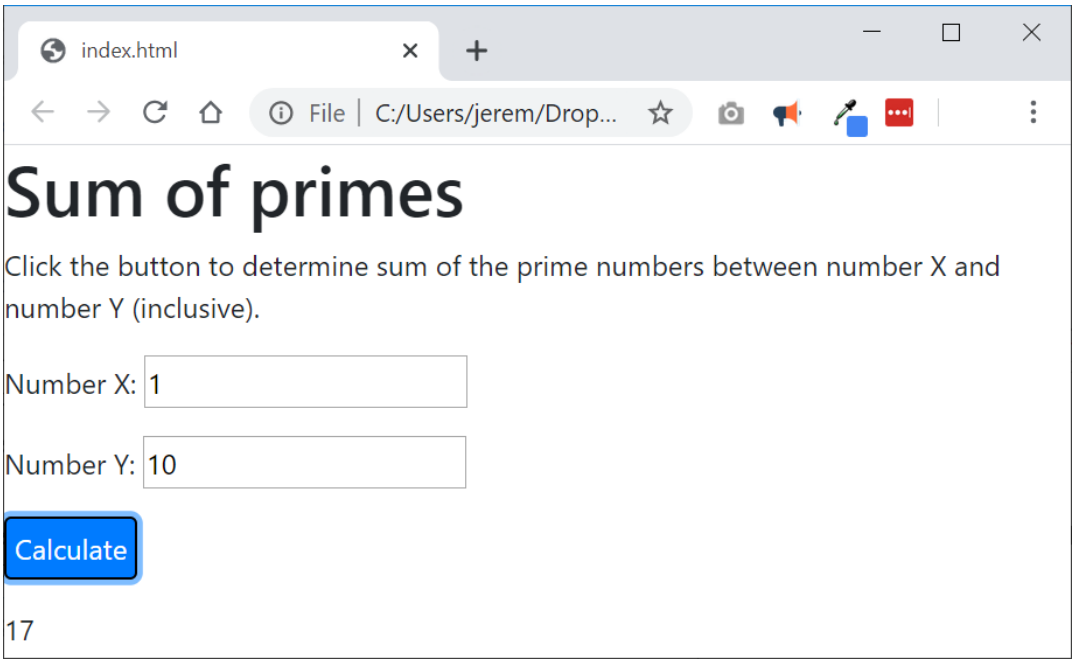
   Here's some logic to help you with the isPrime() function:

   - Let's say that the input into the function is x.
   - If x is one, the function should return false. The function can just stop as soon as it sees the number one. No need to go any further!
   - If x is not one, then we need a loop. The loop should have a counter variable that starts at 2 and continues to increment that counter for as long as the counter variable is **less than** x
   - If x **modulus** the counter is zero, that means that you have found a positive divisor. If you should find a positive divisor, the function can return false, and just stop. No need to continue, you have proven that x is not prime.
   - If the loop runs all the way to the end… guess what? There were no positive divisors between 1 and x (exclusive) and therefore… x must be prime. Return true!

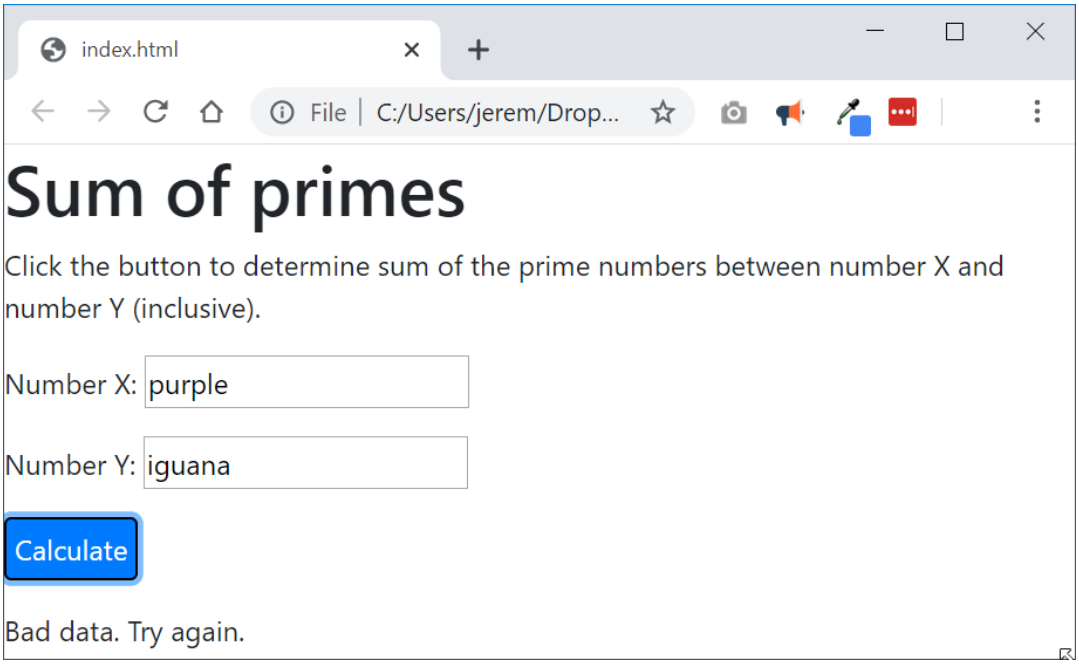
8. Test your work using the web developer console. Changing the input to the isPrime() function should change its output. An example follows:



9. Complete the sumOfPrimes function. Notice that sumOfPrimes takes two parameters, numx and numy. In this assignment, it important to ensure that both numx and numy are integers. That is why the lines of code using parseInt() have been included.
10. Test your work. A sample screenshot is shown below.



11. Test your work. Don't forget to do the error trapping in sumOfPrimes necessary to generate the "Bad data. Try again." message.



12. Upload your work to assignment04 to the corresponding canvas assignment.

How will this assignment be graded?

- If your work generates **all output** correctly, you will get a score of 100%.
- If your work generates **almost all output** correctly (**only one** bad output), you will get a score of 80%
- If your work generates **some output** correctly (some right output, some wrong output), you will get a score of 60%
- If your work generates **only one output** correctly, you will get a score of 40%
- If your work does not generate any correct output, you will get a score of **zero**.