CS/INFO 3300 // INFO 5100
Homework 1
Due at 11:59pm Wednesday, August 30

Goals: Review Javascript types and program flow.

Your work should be in the form of an **HTML file called `index.html` with one <p> element per problem**. Wrap any Javascript code for each problem in a **`<script>`** element nested within the **<p>** element.

For example:

```
<html><body>
  <p id="p0">
    Problem 0: We use a "let" statement to...
    <script>
      let x = 100;
    </script>
  </p>
</body></html>
```

Create a zip archive containing this file and upload it to CMS before the deadline.

1. For each of the following items in the list, please:
A: Identify whether it is a **canonical Javascript type** (i.e., officially documented as a type)
B: If the type is a canonical Javascript type, please **provide an example** in your `<script>` section first using `console.log( ___ )` to print an example and then another call with `console.log(typeof( ___ ))` to show that your example is of the correct type.

a) avocado;	b) array;	c) boolean;	d) object;	e) Unicode;	f) string;
g) function;	h) int;	i) number;	j) class;	k) regex;	l) char;
m) undefined;	n) pointer;	o) NaN (“not a number”);			

2. Null is a canonical Javascript type, but it has some unusual or unexpected `typeof()` behavior. In 1-2 sentences, explain the unusual behavior that it has.

3. First, in 1-2 sentences, please **define type coercion**. Then, in your **<p>** tag please provide two examples where type coercion **produces an answer that is unusual or unexpected** (i.e., different from Python output or a case where Python would throw an exception). For each sample, briefly explain why it is unexpected. Then, in the **`<script>`** section put both of your code examples into a `console.log()` statement so we can see the result (e.g., `console.log("apple" + "orange")`)

(next page)

4. In your `<script>` tag, **create a recursive function, `fib(n)`, which takes an integer, `n`, as an argument and returns the n^{th} number in the Fibonacci sequence**. For example, `fib(4)` would return 2, as 2 is the 4^{th} number in the sequence.

For a starting point, we can assume that `fib(0)=0` and `fib(1)=1`.
To compute the Fibonacci sequence, you need to add up the two numbers prior to the n^{th} item in the list. For example, `fib(4)` is found by adding up the results of `fib(3)` and `fib(2)`. `fib(3)` is the result of `fib(2)+fib(1)`. And so on.

Do not use loops or any built-in Fibonacci functions to accomplish this. Remember that recursion involves two major components: **base case(s)** and **recursive calls** of the function.

Make sure that your `fib` function **handles negative integers by returning `undefined`**, properly **returns `0` for `fib(0)`, `1` for `fib(1)`**, and otherwise **uses recursion to compute correct answers** for larger integers.

Include the following code block after your function to prove it works as expected:

```
console.log( fib(-2) );
console.log( fib(0) );
console.log( fib(1) );
console.log( fib(4) );
console.log( fib(7) );
console.log( fib(9) );
```

(As the function is only intended to handle integers, do not worry about implementing an algorithm to handle decimal numbers)

5. Describe, in your own words, the course policy on late and unreadable work outlined in the syllabus, which is linked from the course website.

Addendum: Brief Check for Pre-req Knowledge

Here are a few questions you can use to help guide whether you have the necessary background knowledge for INFO3300/5100 if you have not taken CS2110 and INFO2300.

Note that this is not intended to be a formal exam, no bonus credit is offered, and doing well on these questions is NOT a guarantee that you will be able to master all course content (just as missing some of the questions does not guarantee you'll do badly in the class). We will also not be providing an answer key. It is offered as a quick way to provide additional security if you are coming to the class from another university or without expected pre-reqs.

Do not include answers to these in your submission. They will not be graded.

1. Define user-centered design. What steps are involved in the user-centered design process? How do we make sure it is rigorous and thorough?
2. Create a CSS selector for any <p> tags that have the class "nell". Now, create one that selects a element that is *inside* of another that has the id "corn".
3. What does the "position" style attribute do to a element? What are the best practices related to positioning elements on the web.
4. Define responsive web design.
5. What does DOM mean in the context of web development?
6. What takes precedence: a style attribute on a specific DOM element, or a CSS style applied to that element in the <head> of the document?
7. Is the DOM a tree, array, or dictionary data structure? Define parent and child nodes in the context of web development.
8. What's the difference between <head> and <body> tags in HTML?
9. Is this valid HTML:
10. What color is #00CC00?
11. How role does PHP play in a web stack?
12. How are data transmitted across the web, and what is the difference between a GET and a POST request?
13. Explain the difference between a list and map. What are the trade-offs of using these data structures, and what are some common use cases of each?
14. What syntax do you use to make maps and lists in Python? What kind of data structure is an array? What kind of data structure is a dictionary?
15. What are the *keys* and *values* in a map?

16. Define *pointer* in the context of object-oriented programming. What are pointers used for, and why are they important for modern programming practices?
17. What is a shallow copy? How does it compare to a deep copy?
18. What does *scope* mean in the context of programming? What's the difference between *block scope* and *global scope*?
19. What is "big O notation"? What is the big O of searching an unordered list versus a well-designed hash-map?
20. What are nodes and edges in a graph? What are the most common ways to represent a network/graph in data structure form?
21. What's the difference between a tree and a graph?
22. Define recursion. What are the requirements for recursion to work?
23. Define loops. What are the requirements for a loop to work? What is the difference between a while loop and a for loop.
24. What is a tree traversal? What's the difference between a pre-, in-, and post-order traversal?
25. What is a *class*, and why is it useful in object-oriented programming?
26. What is the difference between a method and a function. What is a function/method signature? What does it mean for a function to "return"?
27. Explain what *break* and *continue* mean in the context of loops.
28. What is an exception? What do *try* and *except* do in Python?
29. What does a *comparator* do when sorting a list?
30. How do you go about debugging a program? What's the best strategy for interpreting the stacktrace provided when an exception is reached?
31. Why would you use version control systems such as Github? How do you use them to keep a project organized. What is a *commit*, a *branch*, and a *merge conflict*?