Course:	**INFO6144**
Professor:	**Jim Cooper**
Project:	**Lab #4**
Due Date:	**Tuesday, March 5, 2024 11:30 pm**
Submitting:	**Submit your zipped solution to the Lab 4 drop-box**

How will my lab be marked?

Marks Available	How are the Marks Awarded?	Mark Assigned
3	Web page runs and meets the requirements listed below	
1	Reasonable level of comments	
1	Web Page is zipped and submitted on-time to the drop box	
5	Total	

Lab Description

This lab will require you to add some new code to the Lab4.html web page. As we've often done in the past, rename the Lab4.html file to "your first name_lab4.html" for example "Jim_lab4.html"

The purpose of the lab is to create a proof of concept for a simple logging system that can receive calls from multiple asynchronous "timers." We'll use the JavaScript "setInterval" method to create timers that can call a function at various times.

Requirements:

Based on the source code listing shown on the next page, notice that we're starting with:

1. A single global array called "timers."

2. An IIFE (Immediately Invoked Function Expression) called "log"

3. A function literal called "printFunc" which is a "wrapper" function used to output a string (followed by a "
") with "document.write."

4. An "onload" event handler called "startMeUp" which we use to initialize 3 sample timers using the "setInterval" method. Remember that "setInterval" requires 2 parameters (a function reference to be called and an interval in milliseconds to define the frequency with which the function will be called). Also remember that "setInterval" returns a unique integer number that can be used to perform a "clearInterval" on a specific timer.

In order to complete the lab, you must add some code to the IIFE starting after line 14.

1. Return an anonymous function declaration (see ClosureExample04.html). Your anonymous function should define a single parameter named caller which represents the particular timer (Timer1, Timer2 etc.).

2. Within the anonymous function declaration, you will have access to all the parameters and local variables of the parent function. As such, on each invocation:

a. Increment the counter by 1
b. Push the name of the caller into the “callerLog” array
c. Push the current date/time into the dateTime array (use new Date() to retrieve the date/time).
d. Create an output string by concatenating the counter, the last entry in “callerLog” and the last entry in “dateTime.”
e. Output the string using the designated output function. Note you may not use “printFunc” directly.
f. When the counter exceeds 10, shutdown all the timers using the “clearInterval” method.

Screen shot of the initial code:

```
<html>
    <head>
        <title>Lab 4</title>
        <script language="JavaScript" type="text/JavaScript">

        let timers = [];

        let log = (function (outputFunc) {
            let counter = 0;
            let callerLog = [];
            let dateTime = [];

            // return inner anonymous function
            return function(caller) {
                // replace the next line with your new code
                console.log(caller);
            }

        })(printFunc); //IIFE

        function printFunc(output) {
            document.write(output + "<br>");
        }

        function startMeUp() {
            // add each of the timer references to the timers array
            // as part of invoking the log function following each interval
            timers.push(setInterval("log('Timer1')", 1000));
            timers.push(setInterval("log('Timer2')", 1200));
            timers.push(setInterval("log('Timer3')", 1700));
        }

        </script>
    </head>
    <body onload="startMeUp();">

    </body>
</html>
```

Sample Output: