

CSE 3302: Programming Languages Lab

Functional Programming using JavaScript

INSTRUCTIONS

1. Do NOT plagiarize.
2. No group work. All work should be your own.
3. Do not discuss your work with other students in the class.
4. You CANNOT borrow code from online sources.
5. Your program should not ask for any user input.
6. Display your results for each question in a new line.
7. DO NOT submit an HTML file
8. Turn in your program using Canvas. Do not email your program to the TA or the instructor.
9. Name your document as lab02-<netid>.js where <netid> is your UTA netid. If you do not know your netid, check what it is using NetID Self Service. Your 1000 number is NOT your netid.
10. All code should be your own. You may not copy code from the slides, book, others, or the internet unless specified. You are not allowed to use in-built functions other than the ones taught in class for functional programming.
11. Write an explanation of your code for each line using comments. If the explanation is not clear, you will NOT receive full credit.
12. The code should have your name, 1000 number, and the date it is due as the first 3 lines in order.
13. Use the Developer mode of your browser to access the JavaScript command line. You can edit your code in a separate file and then just paste it into the command line to run it. You will be submitting the file with JavaScript.
14. Link used in class is below. This is the link to the first part. There are 6 parts and you can get to other parts from this link: -
<https://medium.com/@cscalfani/so-you-want-to-be-a-functional-programmer-part-1-1f15e387e536>

--- LAB 01 ---

1. (5 points) Start with an array called **inputtable**. The array should have numbers between 1 and 10.

NOTE: Do NOT use a form of a 'for' loop anywhere, including iterators. This is meant to be a functional exercise, so your code is expected to not have side effects.

2. (30 points) Use **inputtable** from step 1 to create the following: -
 - a. Set of multiples of 5 between 1 and 51. Name it **fiveTable**
 - b. Set of multiples of 13 between 1 and 131. Name it **thirteenTable**
 - c. Set of squares of the numbers in **inputtable**. Name it **squaresTable**

3. (10 points) Get the odd multiples of 5 between 1 and 100. 5, 15, ...
4. (20 points) Get the sum of even multiples of 7 between 1 and 100.
 - a. Example, find the multiples and then sum them: 14 + 28+...
5. (15 points) Use currying to rewrite the function below: -

```
function cylinder_volume(r, h){  
    var volume = 0.0;  
    volume = 3.14 * r * r * h;  
    return volume;  
}
```

 - a. Use r = 5 and h = 10 to call your curried function.
 - b. Reuse the function from part 'a' but use h = 17
 - c. Reuse the function from part 'a' but use h = 11
6. (15 points) Use the following code to take advantage of closures to wrap content with HTML tags, specifically show an HTML table consisting of a table row that has at least one table cell/element. You can use the console to output your results.

```
makeTag = function(beginTag, endTag){  
    return function(textcontent){  
        return beginTag +textcontent +endTag;  
    }  
}
```

Example output for #6. Note that the <th> tag is optional. Please do not use this data, but substitute your own values for the contents of the cells.

```
<table>  
<tr>  
  <th>Firstname</th>  
  <th>Lastname</th>  
  <th>Age</th>  
</tr>  
<tr>  
  <td>Jill</td>  
  <td>Smith</td>  
  <td>50</td>  
</tr>  
<tr>  
  <td>Eve</td>  
  <td>Jackson</td>  
  <td>94</td>  
</tr>  
</table>
```

7. (5 points) Following instructions
8. (Extra credit) Do the 'generic' version of questions 3 and 4, meaning the target multiple must not be hard coded. This means you should be able to use the same code to handle multiple scenarios, for example: first odd multiples of 11 and then even multiples of 3 (still in the range 1 to 100).