

# CMSI 185-01

## COMPUTER PROGRAMMING

Fall 2014

### Assignment 113

Not surprisingly, much of this assignment focuses on functions. Even the first of the two pre-Chapter 5 questions benefits from that knowledge.

#### Outcomes

This assignment ranges the full set of proficiency measures now, from 1 to 5*b*.

#### Background Material

All but the first item are available as links on the course website.

- Dionisio/Toal Chapters 4 and 5
- The Wikipedia article on JavaScript
- The first 5–6 pages of *mtfjs*—test your programming knowledge by seeing how many of the posts you understand...bonus points if you groan or laugh out loud :) (well not really because there are no points in this class, but anyway you get the point right...?)
- The 2010 Google Tech Talk “Building a JavaScript-Based Game Engine for the Web”
- “HTML5 Isometric Game Engine built using JavaScript”—another video about another JavaScript game engine

#### For Submission

Instructions from the syllabus continue to apply, but now for triples rather than pairs. Upload your work to Blackboard/MyLMUConnect as a PDF.

These guidelines for questions that require the submission of code continue to apply:

1. Do your work in jsFiddle and save your links; supply the jsFiddle address in your submission
2. Include your source code, properly formatted, directly in the PDF as well

It is also no longer enough that your programs function correctly. They must also be structured well (2), stick to recommended practices (3), and be presented well (5*a*).

1. Textbook Chapter 4, Problem 20 (2, 3, 5*a*)
2. Textbook Chapter 4, Problem 26 (2, 3, 5*a*)

3. Go through the Codecademy Course *Functions in JavaScript* (link on the course website). On the submitted homework PDF you turn in, simply sign your names next to the phrase “We completed the assigned Codecademy Course” and include a screenshot of the Codecademy page showing the badge you received from completing the course. (1)
4. Textbook Chapter 5, Problem 3 (2, 3, 5*a*)
5. Textbook Chapter 5, Problem 5 (2, 3, 5*a*)
6. Consider what it would take to make the previous program as robust and “foolproof” as possible. Name three special cases for input values that would be worth testing against your function. (4)
7. Textbook Chapter 5, Problem 6 (2, 3, 5*a*)
8. Textbook Chapter 5, Problem 8 (2, 3, 5*a*)
9. Textbook Chapter 5, Problem 16 (2, 3, 4, 5*a*)
10. Textbook Chapter 5, Problem 17 (2)
11. Textbook Chapter 5, Problem 25 (2, 3, 5*a*)
12. Write a list of ten (10) or more test cases that would “exercise” your solution to the above problem for robustness and generality. Try to capture as many combinations as you can see of correct or incorrect values and possible results of the requested functions. For each test case, write the code that tests the case and the result that you expect if your code works correctly. (*Hint*: I can think of at least 15 distinct cases, so no shortage here.) (4, 5*a*)