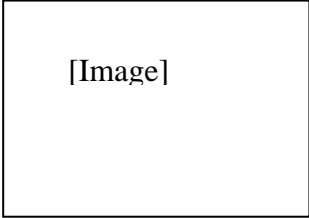


# COMP 690 E-commerce Fall 2007 Programming Assignment 1

Due in the digital drop box by Fri., Aug. 31, at 11:00PM.

To submit your assignment, put all your files (two for Problem 1, three for Problem 2, and one for Problem 3) in a .zip or .rar file, and put that file in the drop box.

1. Create an HTML document in the format shown below

<b>Biographical Summary of [Your Name]</b>									
[Date]									
<table border="1"><thead><tr><th>Topic</th><th>Summary</th></tr></thead><tbody><tr><td><a href="#">Education</a></td><td>...</td></tr><tr><td><a href="#">Interests</a></td><td>...</td></tr><tr><td><a href="#">Career Plans</a></td><td>...</td></tr></tbody></table>	Topic	Summary	<a href="#">Education</a>	...	<a href="#">Interests</a>	...	<a href="#">Career Plans</a>	...	
Topic	Summary								
<a href="#">Education</a>	...								
<a href="#">Interests</a>	...								
<a href="#">Career Plans</a>	...								
<b>Education</b>									
...									
...									
...									
<b>Interests</b>									
...									
...									
...									

Substitute the obvious for the items in square brackets ([ ... ]). The table to the left of the image shown be an HTML `table` element. The contents of the cells on the right should be summaries of the corresponding topics on the left. (For example, the cell immediately below the heading “Summary” should be a summary of your education.) The label “Education” in the first column should be an anchor with an internal link to the header “Education” below, where you go into some detail on you education. Include a bulleted list (use `ul`) of the schools you have attended. The label “Interests” in the table should be an anchor with a link to the header “Interests” below, where you go into some detail about your interests. Finally, the label “Career Plans” in the table should be a hyperlink to another HTML document where you go into some detail about your career plans. Have that document in the same folder and use relative addressing (use only the file name in the URL, not the folder path). The image can be whatever you want; the easiest is to

look at some Web pages and copy an image you like. Your page won't look exactly like the above; you can easily make something that looks nicer. Be sure to use the appropriate header elements (h1, h2, ...).

2. Create an HTML document that presents the initial part of the class syllabus, through the part entitled "Objectives". Use an internal stylesheet (using a `style` element in the document head), but also link to an external stylesheet (a file with extension `.css`). The external stylesheet should import (use the `@import` command) another style sheet (another file with extension `.css`). Define rules for at least four elements that appear in the body of the HTML document. And define at least two classes that are used in the body. Include at least two rules in the internal stylesheet, at least two rules in the external style sheet to which the HTML document links, and at least two rules in the imported stylesheet. Include at least one example where a property value is overridden and at least one example where it isn't. Your version of the syllabus need not look like the above (it should look better), but it should have the same structure and information.

3. Write a JavaScript program that uses prompt boxes to input the number of exams and the number of students. It then uses prompt boxes to input the grades for all the students on all the exams into a two-dimensional array. Exams correspond to rows, students to columns. Assume that, on all exams, scores are integers and can range from 0 to 100. Use a `while` loop to validate each value, warning the user in the re-prompt if (s)he entered a non-numeric value or an out-of-range numeric value. To test whether a text value cannot be interpreted as a number, use the function `isNaN(val)`, which returns `true` if `val` cannot be interpreted as a number. Check whether an entered value is numeric before checking whether it is in range. For numerical comparisons, JavaScript automatically converts a string value to a number, but, before storing a value in the array, convert it to an integer (`parseInt`). The input function, then, has a loop (the validation loop) inside a loop (over the students for each exam) that in turn is inside a loop (over the exams). Output the contents of the array as a table, with rows labeled with the names of exams ("Exam 1", "Exam 2", etc.) and columns labeled with the "names" of students ("Student 1", "Student 2", etc.). Then, for each exam, output the lowest grade anyone received on an exam. See the example program in Section 5.3 that uses a two-dimensional array. That example supplies the array as an initialization value. You should write a separate input function (with the nested loops mentioned above). The following is an example output with three exams and three students.

	<b>Student 1</b>	<b>Student 2</b>	<b>Student 3</b>
<b>Exam 1</b>	60	70	80
<b>Exam 2</b>	90	80	70
<b>Exam 3</b>	65	75	65

The lowest score on Exam 1 was 60.  
The lowest score on Exam 2 was 70  
The lowest score on Exam 3 was 65