Problem 1. (2 x 7 = 14 points)
Consider the following table. Assume that a JavaScript program was loaded from http://www.cornell.edu/dir/page.html. For each row in the following table, both indicate and motivate the result of the JavaScript Same Origin Check.

Example: Under “Result”, you can indicate either succeed or fail. Under “Motivation”, you can say, respectively, same domain, port, and protocol or, for example, different protocol.

<table>
<thead>
<tr>
<th>URL of Target Window</th>
<th>Result</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a <a href="http://www.cornell.edu/index.html">www.cornell.edu/index.html</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b cornell.edu/~soule/index.html</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c ftp://www.cornell.edu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d <a href="http://www.columbia.edu">http://www.columbia.edu</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e <a href="http://www.cornell.edu:80/page1.html">http://www.cornell.edu:80/page1.html</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f <a href="http://www.cornell.edu:8080/page2.htm">http://www.cornell.edu:8080/page2.htm</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g <a href="http://www2.cornell.edu/dir/page.html">http://www2.cornell.edu/dir/page.html</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problem 2. (2 + 4 = 6 points)
Consider the following PHP program instruction:

```
$query = "SELECT * FROM accounts WHERE name='$name' AND password='$password'";
```

This code generates a query intended to be used to authenticate a user who tries to login to a Web site.

a. Show how an attacker can embed a name and password that could cause a table in the database to be erased (2 points).

b. Write a simple sanitization function in PHP that sanitizes name and password before they are used (4 points).

Problem 3. (4 points)
Consider the Cross-Site Scripting (XSS) vulnerability described in Slide 31 of the lecture, which allows a malicious user to embed JavaScript code into what is supposed to be a parameter value. For example, an attacker could cause the parameter value to be

```
John<script>alert('Uh oh');</script>
```

Describe how that vulnerability could be prevented.

Problem 4. (4 points)
Explain why the call to `document.write(b.f)` in Slide 36 is a potential vulnerability.

Problem 5. (4 points)
Explain why the call to `document.write(s)` in Slide 37 is safe or unsafe.

Problem 6. (4 points)
Explain why instruction `el1.innerHTML = el2.innerText` in Slide 38 is unsafe.

Problem 7. (4 points)
Explain why the call to `document.write(a.f)` in Slide 44 constitutes a taint violation, while the call to `document.write(c.f)` does not.