Computer Science 571 2nd Exam
Prof. Papa
Tuesday, April 30, 2013, 5:30pm – 6:40pm

Name: **Student ID Number:**

1. **This is a closed book exam.**
2. **Please answer all questions on the test**

PHP Questions [10 pts]

Each question is worth 2 points.

Q1: What numeric types are supported by PHP?

A1: Integer and floating-point numbers

Q2: Which of the following is true in PHP?

A2:
[X] PHP is not strictly typed
[X] A data type is either text or numeric
[ ] Variables are case-insensitive
[X] PHP decides what type as variable is
[ ] You can use 1 and 0 instead of TRUE and FALSE
[ ] Constant names begin with a dollar sign ($)
[ ] ALL OF THE ABOVE

Q3: What kind of assignment is `$bar = &$foo;`

A3: Assignment by Reference

Q4: Write two statements that concatenate and print the two strings "SEAT" and "Altea":

A4:
```
$car = "SEAT" . "AAltea";
echo $car
```

Q4: Consider this PHP code snippet:

```
<?php
$foo = 25;

echo "5x5=$foo"; // double quotes
echo '5x5=$foo'; // single quotes
```

What is the output produced?

A4:

5x5=25
5x5=$foo

Q5: Assume a form contains one textbox named "txtName", and the form is submitted using the POST method. How would a PHP script access the form data?

A5: $_POST['txtName']

Web Security Questions [10 pts]

Each question is worth 2 points.

Q1: What are "brute force attacks"?

A1: automated processes of trial and error used to guess a person's username, password, session id or authentication cryptographic keys

Q2: What are 2 easy ways to avoid "brute force attacks"?

A2:

1) limit the amount of unsuccessful logins to a small number and then lock out the account

2) block IP addresses where consecutive trial and errors come too quick for a human typist

Q3: What is an easy way to create a password that cannot be looked up in a dictionary?

A3: Use the first letters of the words in a very long sentence

Q4: Give 2 examples of "commonly used" weak passwords

A4: Password1, guest, 123456, letmein, iloveyou

Q5: Name one of the two ways to reduce the threat of Cross-site Scripting (XSS)?

A5:

Any one of these two:

- **Use escaping schemes like HTML entity encoding, JavaScript escaping, CSS escaping and URL or percent encoding**
- **Tie session cookies to the IP address of user that originally logged in and only permit that IP to use the cookie**

HTML5 Questions [10 pts]

Each question is worth 2 points.

Q1: What is the current status of the HTML5 vocabulary, associated APIs and HTML Canvas specifications?

A1: Candidate Recommendations

Q2: Which of the following are new features in HTML5?

[X] publication dates and time
[X] web page sections
[ ] applet upgrade
[X] offline web applications
[ ] network based storage
[X] persistent local storage
[ ] ALL OF THE ABOVE

Q3: What is the benefit of using AAC vs. MP3 codec encoding?

A3: AAC provides support for up to 48 channels of sound (including surround sound) while MP3 only provides 2 channels: left and right.

Q4: What is the meaning of the "autoplay" attribute?

A4: specifies that the video will start downloading and playing as soon as possible after the page loads.

Q5: Name two popular video codecs?

A5: any 2 of H.264, Theora, VP8 (aka WebM), Sorenson Spark

Java Servlets Questions [10 pts]

Below is the Java source which implements a portion of the proxy back-end of Homework #8, but some of the code is missing, replaced by XXXXXXXs. Fill in the missing code. Each answer is worth 1 point.

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

public class MusicSearch extends HttpServlet {


      public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            PrintWriter out = response.getWriter();
        try {

            String title = request.getParameter("title");
```

```
            String type = request.getParameter("type");
                  ---------------------------------------------------------------------

            title = title.replace(' ','+');
                  ---------------------------------------------------------------------

            response.setContentType("text/html; charset=UTF-8");

                  // Retrieve xml from perl script
            String data = getPerlData(title,type);
                  ---------------------------------------------------------------------

            //out.println(data);

            //Call appropriate parsing function depending on type
            String json = "";
            if(type.equals("artists")) {
                        json = parseArtists(data);
                  } else if(type.equals("albums")) {
                        json = parseAlbums(data);
                  } else if(type.equals("songs")) {
                        json = parseSongs(data);
                  }

            //Return json back to html
            out.println(json);
          } catch(Exception e) {
                  ---------------------------------------------------------------------
              System.out.println(e.getMessage());
          }
        }

        public String getPerlData(String title, String type) throws Exception
{

          URL url = new URL("http://cs-server.usc.edu:18493/cgi-
bin/music_xml.pl?title="+title+"&type="+type);
          URLConnection connection = url.openConnection();
                  ---------------------------------------------------------------------

          BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream(), "UTF-8"));
                  ---------------------------------------------------------------------

          String xmlString;
          String totalString="";
          while((xmlString = in.readLine() )!=null)
                  ---------------------------------------------------------------------

            {
              totalString += xmlString;
            }
          in.close();
```

```
        return totalString;
      }

      public String parseArtists(String data) throws JDOMException,
IOException {

[... CODE REMOVED ...]
      }

      public String parseAlbums(String data) throws JDOMException,
IOException {
[... CODE REMOVED ...]
      }

      public String parseSongs(String data) throws JDOMException,
IOException {
[... CODE REMOVED ...]
}
```

Web Performance Questions [10 pts]

Each question is worth 2 points.

Q1: What happens when the following header is present in a response?

```
cache-control: max-age
```

A1: the response is stale if its current age is greater than the age value given (in seconds) at the time of a new request for the resource

Q2: What does the presence of the `max-age` directive implies?

A2: It implies that the response is cacheable

Q3: Why is the use of CSS sprites beneficial to the overall performance of a web page?

A3: Using images sprites reduces the number of server requests and saves bandwidth

Q4: Why is it better to use GET instead of POST in AJAX requests?

A4: It is better to use GET instead of POST since GET sends the headers and the data together, while POST sends the header and the data separately.

Q5: For performance, is it beneficial to send an image with dimensions "smaller" than the width and height attributes of the HTML image element that will contain it?

A5: No, because the browser still needs to perform "expansion" of the image at run time to fit the new dimensions.

JSON Questions [10 pts]

Recently Yahoo provides access to the Flickr API using YQL, the Yahoo! Query Language. For example, the YQL query to "Get user info from Flickr ID", looks like this:

select * from flickr.people.info2 where user_id="26545327@N00" and api_key="92bd0de55a63046155c09f1a06876875";

Results can be requested in JSON or XML. The XML REST call is:

http://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20flickr.people.info2%20where%20user_id%3D%2226545327%40N00%22%20and%20api_key%3D%2292bd0de55a63046155c09f1a06876875%22%3B&diagnostics=true

And the result returned is:

```
<?xml version="1.0" encoding="UTF-8"?>
<query xmlns:yahoo="http://www.yahooapis.com/v1/base.rng"
    yahoo:count="1" yahoo:created="2013-04-27T19:55:50Z" yahoo:lang="en-US">
    <diagnostics>
        <publiclyCallable>true</publiclyCallable>
        <user-time>182</user-time>
        <service-time>179</service-time>
        <build-version>36288</build-version>
    </diagnostics>
    <results>
        <person datecreate="1132792566" iconfarm="1" iconserver="30"
            id="26545327@N00" ispro="0" nsid="26545327@N00"
path_alias="fabiokung">
            <username>Fabio Kung</username>
            <realname>Fabio Kung</realname>
            <location>São Paulo, Brazil</location>
            <timezone label="Brasilia" offset="-03:00"/>
            <description>&lt;a href="http://fabiokung.com/about"
rel="nofollow"&gt;fabiokung.com/about&lt;/a&gt;</description>
            <photosurl>http://www.flickr.com/photos/fabiokung/</photosurl>
            <profileurl>http://www.flickr.com/people/fabiokung/</profileurl>

<mobileurl>http://m.flickr.com/photostream.gne?id=1805705</mobileurl>
            <photos>
                <firstdatetaken>2003-01-01 00:00:01</firstdatetaken>
                <firstdate>1142015714</firstdate>
                <count>175</count>
            </photos>

<buddyiconurl>http://farm1.static.flickr.com/30/buddyicons/26545327%40N00.jpg</buddyiconurl>
        </person>
    </results>
</query>
```

The JSONP REST call is (notice &format=json and &callback=cbfunc):

http://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20flickr.people.info2%20where%20user_id%3D%2226545327%40N00%22%20and%20api_key%3D%2292bd0de55a63046155c09f1a06876875%22%3B**&format=json**&diagnostics=true**&callback=cbfunc**

Complete the missing parts of the JSONP result:

```
cbfunc({
 "query": {
```

```
  "count": 1,
  "created": "2013-04-27T19:58:46Z",
  "lang": "en-US",
  "diagnostics": {
   "publiclyCallable": "true",
   "user-time": "186",
   "service-time": "182",
   "build-version": "36288"
  },
  "results": {
   "person": {
    "datecreate": "1132792566",
    "iconfarm": "1",
    "iconserver": "30",
    "id": "26545327@N00",
    "ispro": "0",
    "nsid": "26545327@N00",
    "path_alias": "fabiokung",
    "username": "Fabio Kung",
    "realname": "Fabio Kung",
    "location": "São Paulo, Brazil",
    "timezone": {
     "label": "Brasilia",
     "offset": "-03:00"
    },
    "description": "<a href=\"http://fabiokung.com/about\"
rel=\"nofollow\">fabiokung.com/about</a>",
    "photosurl": "http://www.flickr.com/photos/fabiokung/",
    "profileurl": "http://www.flickr.com/people/fabiokung/",
    "mobileurl": "http://m.flickr.com/photostream.gne?id=1805705",
    "photos": {
     "firstdatetaken": "2003-01-01 00:00:01",
     "firstdate": "1142015714",
     "count": "175"
    },
    "buddyiconurl":
"http://farm1.static.flickr.com/30/buddyicons/26545327%40N00.jpg"
   }
  }
 }
});
```

XML Schema Question [10 pts]

Google Sitemap uses the Sitemap Protocol to allow webmasters to inform search engines about URLs on websites that are available for crawling. In its simplest form, a Sitemap that uses the Sitemap Protocol is an XML file that lists URLs for a site.

Google provides an XML Schema that describes the structure of these files, as shown below (some optional elements have been removed for simplicity):

```
<xsd:schema xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
targetNamespace="http://www.sitemaps.org/schemas/sitemap/0.9"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
```

```
    <xsd:documentation>
    XML Schema for Sitemap files.
    Last Modifed 2006-07-25
  </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="urlset">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" ref="url" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="url">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="loc" />
        <xsd:element minOccurs="0" ref="changefreq" />
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="loc">
    <xsd:annotation>
      <xsd:documentation>
      REQUIRED: The location URI of a document.
      The URI must conform to RFC 2396
(http://www.ietf.org/rfc/rfc2396.txt).
    </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleType>
      <xsd:restriction base="xsd:anyURI">
        <xsd:minLength value="12" />
        <xsd:maxLength value="2048" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="changefreq">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="always" />
        <xsd:enumeration value="hourly" />
        <xsd:enumeration value="daily" />
        <xsd:enumeration value="weekly" />
        <xsd:enumeration value="monthly" />
        <xsd:enumeration value="yearly" />
        <xsd:enumeration value="never" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
</xsd:schema>
```

Create an instance XML file that contains two "url" entries:

1. **The first entry for `http://www.example.com/` , has changefreq = daily;**
2. **The second entry for `http://www.example.com/catalog?item=12&desc=vacation_hawaii` has changefreq = weekly.**

3. Notice that the namespace id of the root element is
`http://www.sitemaps.org/schemas/sitemap/0.9`

```
<?xml version="1.0" encoding="UTF-8"?>

<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

   <url>

      <loc>http://www.example.com/</loc>


      <changefreq>daily</changefreq>


   </url>

   <url>


<loc>http://www.example.com/catalog?item=12&amp;desc=vacation_hawaii</loc>

      <changefreq>weekly</changefreq>

   </url>
</urlset>
```

JSON/AJAX Questions [10 pts]

Q1: The open() method of the XMLHttpRequest object has this syntax:

open("method", "URL", "flag", "username", "password")

with the last three parameters being optional. What is the purpose and the default value of the third parameter, "flag"?

A1: The third parameter is a boolean value indicating whether or not the request will be asynchronous; the default value of this parameter should be assumed to be true.

Q2: What are basic technologies used in AJAX?

A2:

[ ] HTML
[ ] CSS
[ ] JavaScript
[ ] XML
[ ] JSON
[ ] XHTML
[ ] XSLT
[ ] Web Remoting

[ ] DOM
[X] ALL OF THE ABOVE

Q3: What is the major problem with the "classic" web application model vs. the "Ajax" application model?

A3: User actions trigger synchronous requests to the server, and while the server is doing its things, the user waits

Q4: List 3 properties of JSON What method of the XMLHttpRequest object is used to retrieve JSON returned data?

A4: responseText

Q5: What is the following:

```
(/^(\s|[,:{}\[\]]|"(\\["\\bfnrtu]|[^\x00-\x1f"\\])*"|-
?\d+(\.\d*)?([eE][+-]?\d+)?|true|false|null)+$/.test(text))
```

A5: A regular expression used to parse JSON and ensure that it is safe

Cookies and Privacy Questions [10 pts]

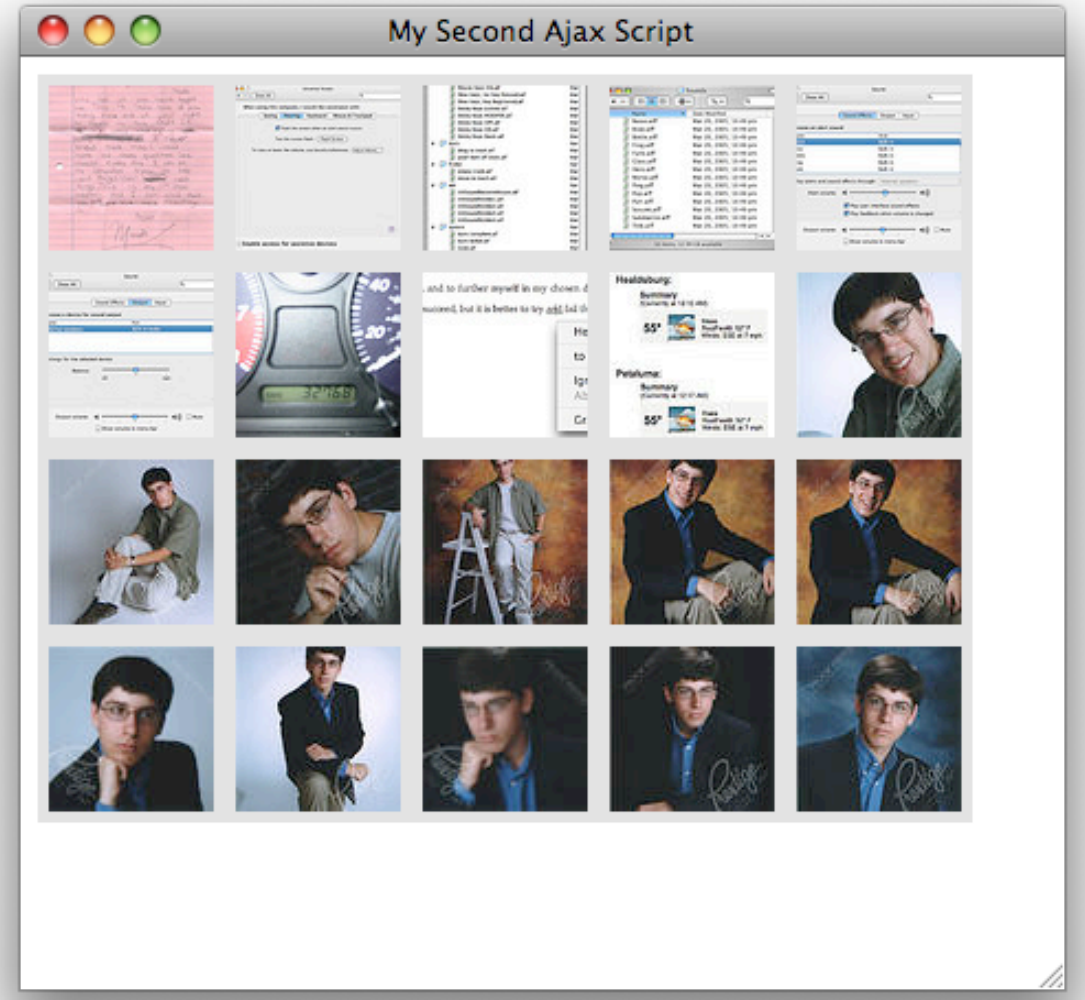
Q1: Write a JavaScript function that creates a cookie with a given value and expiration date. Use the API toGMTString() to produce a date in the correct format. You do not have to check for the validity of the passed parameters.

```
function setCookie(name, value, expireDate) {

  expString = ";expires=" + expireDate.toGMTString();
  document.cookie = escape(name) + "=" + escape(value) + expString + "; ";

}
```

Q2: Write a JavaScript function that retrieves the value of a cookie, given its name. Complete the given function below.

```
function getCookie(name) {
      var item = unescape(name) + "=";
      if (document.cookie.length > 0) {
            posit = document.cookie.indexOf(item);
            if (posit != -1) {
              posit = posit + item.length;
              last = document.cookie.indexOf("; ", posit)
              if (last == -1)
                        last = document.cookie.length;
              return unescape
      (document.cookie.substring(posit,last));
      }
  }
}
```

JavaScript and Ajax Questions [10 pts]



Below is the HTML source code that produces the web page above.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
<html>
<head>
      <title>My Second Ajax Script</title>
      <script src="script02.js" type="text/javascript"
language="Javascript">
      </script>
</head>
<body>
<div id="pictureBar"> </div>
</body>
```

```
</html>
```

Below is the JavaScript source code, script02.js, that was imported into the HTML above, but some of the lines are missing, replaced by XXXXXXXs. Fill in the missing.

```
window.onload = initAll;
var xhr = false;

function initAll() {
	if (window.XMLHttpRequest) {
		xhr = new XMLHttpRequest();
	}
	else {
		if (window.ActiveXObject) {
			try {
				xhr = new ActiveXObject("Microsoft.XMLHTTP");
			}
			catch (e) { }
		}
	}

	if (xhr) {
		xhr.onreadystatechange = showPictures;
		xhr.open("GET", "flickrfeed.xml", true);
		xhr.send(null);
	}
	else {
		alert("Sorry, but I couldn't create an XMLHttpRequest");
	}
}

function showPictures() {
	var tempDiv = document.createElement("div");
	var pageDiv = document.getElementById("pictureBar");

	if (xhr.readyState == 4) {
		if (xhr.status == 200) {
			tempDiv.innerHTML = xhr.responseText;
			var allLinks = tempDiv.getElementsByTagName("a");

			for (var i=1; i<allLinks.length; i+=2) {
				pageDiv.appendChild(allLinks[i].cloneNode(true));
			}
		}
		else {
			alert("There was a problem with the request " +
xhr.status);
		}
	}
}
```