ECS 189H
Web Programming
Practice Midterm 2 – Spring 2017

1. There are a number of things the browser can do that the server cannot, and also a number of things the server can do that the browser cannot. Label each Javascript line below as most likely browser code (B), server code (S), or generic code (G) that could be in either the browser or server. Add a sentence of explanation if you think the answer is ambiguous.
   - `var request = new XMLHttpRequest();` - B
   - `url = "http://apis.instagram.com/api?getPhoto=509ad80b"` - G
   - `response.send(outputString);` - S
   - `this.idNum = id;` - G
   - `exports.makeElement = makeElement;` - S
   - `var newObj = JSON.parse(newData);` - G
   - `document.getElementById('map')` - B

2. Answer True or False. Add a sentence of explanation only if you think the question is ambiguous.

   a) Anonymous functions have to be defined inside other functions. - F

   b) The Same Origin Policy is implemented by the browser. - T

   c) An AJAX request should always define a callback. - T

   d) A server can send an HTTP GET request to a browser. - F

3. Multiple choice: Which of the following statements about our Photobooth project is true? Add a sentence of explanation only if you think the question is ambiguous.
   a) We sent an HTTP request from the browser to the Google Cloud Vision API using an XMLhttpRequest object. - F
   b) We sent an HTTP request from the server to the Google Cloud Vision API using code from a Node module. - T
   c) We sent an HTTP response from the server to the Google Cloud Vision API using the response object. - F

4. Rewrite the following function so that it does not use an anonymous function, but still runs correctly.

```
function answer(query, response) {
    db.run(
        'INSERT OR REPLACE INTO photoLabels VALUES (?, "", 0)',[query.fn, query.labels],
            function () {
                response.send("Stored labels "+query.labels+" for "+query.fn);
            }
      );
}

Answer:

function answer(query,response) {
    db.run(
         'INSERT OR REPLACE INTO photoLabels VALUES (?, "", 0)',[query.fn, query.labels],
         dbCallback );

     function dbCallback() {
         response.send("Stored labels "+query.labels+" for "+query.fn);
      };
 }



\item
Write out the contents of x, as a Javascript literal.  What is its data type?
\begin{verbatim}
function showItemPrice(str) {
    console.log(str);
    }

function createAction (index, price) {
    return function() {
        showItemPrice("Item "+index+" costs "+price);
    }
}

x = createAction(5, "$37.99");
```

Notice that x has to be a function, indeed the anonymous function that appears in the code. A ”literal” is the Javascript syntax for writing down an object (eg. ”2” is an integer literal, ”{ animal: 'cow' }” is an object literal, and ”function() { return 2; }” is a function literal. Also, because the function is defined in the closure of createAction, it knows the values of ”index” and ”price”.

```
Answer:

x = function () {
            showItemPrice("Item 5  costs $37.99");
    };
```

5. In an effort to provide some privacy for photos on our server, we store a photo not under its real name but using a 10-digit ID number, which is stored in the database. So when the browser wants to display a photo, instead of `hula.jpg` it needs to know its ID number; for instance it would access hula.jpg as

   `138.68.25.50:????/7563204921.jpg`

   The browser can find out the ID number of a photo using an AJAX request to the server. We have decided that the form of this query should be something like this:

   `138.68.25.50:????/query?op=photoID&filename=hula.jpg`

   Assume that your colleagues have already written the server code to correctly handle this operation; it returns the ID number in the body of the response, or the string ”not found” if the photo is not in the database.

   To test this out, we'll make a Web page where the user can type in the name of a photo they uploaded, the browser will make a query to the server to find out its ID number, and then it will display the photo. The code will be run in response to a button click on the Web page, and the HTML for the button and the text box looks like this:

```
<input type="text" id="photoQuery">
<button id="submit" onclick="getPhotoID()">submit</button>
```

   Our Web page (for now) will do nothing when the photo is not found, but it should work when the photo is found on the server.

   Fill in the missing lines so that it runs successfully.

```
function getPhotoID() {
    var url = "http://138.68.25.50:????";

    var filename = document.getElementById('photoQuery').value;
    var oReq = new XMLHttpRequest();

 //ANSWER:

     oReq.open(GET, url+"/query?op=photoID&filename="+filename);

     // define callback
     oReq.addEventListener("load", responseCallback);
     function responseCallback() {
          displayPhoto(this.responseText;
          }

     oReq.send();  // sends it off
}

function displayPhoto(idNum) {
    var body = document.getElementByType("body");

    var newDiv = document.createElement("div");
    newDiv.id = "photoDiv";

    var newImg = document.createElement("img")

// ANSWER:

    newDiv.appendChild(newImg);
    newImg.src = "http://138.68.25.50:????/"+idNum+".jpg";


    body.appendChild(newDiv);
}
```