St. Lawrence University
Math, Computer Science & Statistics
CS 332: Web Programming
Fall 2021

Practice Exam 1

Tuesday, October 12, 2021
10:10am – 11:40am (90 minutes)

Username: _____ **Student ID:** _____

Name: _____

Honor Pledge:
"I have neither given nor received unauthorized aid on this examination,
nor have I concealed any violations of the Honor Code."

Signature: _____

Instructions:

- The exam is closed book and notes except for one 8.5"x11" sheet of notes (both sides). All electronic device use is prohibited during the exam (calculators, phones, laptops, etc.).
- Print your name, student ID number, and Username **LEGIBLY**. Sign the Honor Pledge. Your exam will not be graded without your signature.
- Record your **USERNAME** at the top of each odd-numbered page in the space provided. This will allow us to identify your exam if pages become separated during grading.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Partial credit will be awarded for partial solutions. If you leave a non-extra-credit portion of the exam blank, **you will receive one-third of the points for that small portion (rounded down)**. If you wish to make an answer to a sub-question blank, draw an "X" through the portion you wish to be blank. **DO NOT** cross out portions of the exam until you near the end because this mark cannot be undone.
- Partial credit will be awarded for partial solutions.
- **DO NOT** detach any pages from this booklet.
- There should be 10 pages in this packet; if you are missing any pages, please let the instructor know.

---

1. **Web Architecture** [11 Points]

Consider the following output from the **client URL** (cURL) program for an HTTP request to `google.com`.

```
% curl -v google.com
*   Trying 142.250.176.206...
* TCP_NODELAY set
* Connected to google.com (142.250.176.206) port 80 (#0)
> GET / HTTP/1.1
> Host: google.com
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Location: http://www.google.com/
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 07 Oct 2021 00:00:17 GMT
< Expires: Sat, 06 Nov 2021 00:00:17 GMT
< Cache-Control: public, max-age=2592000
< Server: gws
< Content-Length: 219
< X-XSS-Protection: 0
< X-Frame-Options: SAMEORIGIN
<
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
* Connection #0 to host google.com left intact
* Closing connection 0
```

Label the key parts of the request and response. Explain what each part signifies in the space below.

2. Web Applications [16 Points]

A) (8 points) List four properties or characteristics often exhibited by modern web applications.

1.

2.

3.

4.

B) (8 points) Identify a currently popular web application and describe how the properties you listed in the previous part apply to this application.

Web Application: _____

1.

2.

3.

4.

3. **HTML5** [13 Points]

For each of the following tags, describe the type of information typically contained within.

A) `<article>`

B) `<aside>`

C) `<div>`

D) `<figure>`

E) `<footer>`

F) `<header>`

G) `<li>`

H) `<nav>`

I) `<ol>`

J) `<p>`

K) `<span>`

L) `<strong>`

M) `<ul>`

4. **Accessibility** [5 Points]

Explain the purpose of `aria` attributes in HTML tags. Give one concrete example and describe its impact on the web browser.

5. The DOM and Cascading Style Sheets [10 Points]

A) (8 points) Draw the Document Object Model (DOM) that would be generated from the following snippet of HTML. Include all opening tags from the code snippet in your drawing. You may omit the text contained within the HTML elements.

```
<div>
    <h1>CS 332</h1>
    <h2>Topics <span>in this class</span></h2>
    <ul>
        <li>Web Architecture</li>
        <li>
            Content Display and Styling
            <ol>
                <li>HTML5</li>
                <li>CSS</li>
            </ol>
        </li>
    </ul>
</div>
```

B) (3 points) Write a CSS rule that changes the color of the list item containing HTML5 (and only HTML5) in the document above to **red**.

6. Positioning [10 Points]

A) (5 points) Explain how to position DOM elements next to each other using Flexbox layout. How can you make this layout *responsive*?

B) (5 points) Describe two similarities and two differences between *relative* and *absolute* positioning.

7. Higher-Order Functions [20 Points]

A) (10 points) Write a function, `multiplyEvens`, in JavaScript that takes in an array of numbers as an argument and returns the product of the elements whose values are divisible by two. For example, `multiplyEvens([1,2,3,4])` would return `8`. You may use any built-in functions you wish, but this is solvable using only the function provided on the last page of this exam.

```
function multiplyEvens(arrayA) {




}
```

B) (10 points) Write a function, `insertListItems`, in JavaScript that takes in a DOM element, and an array of strings and inserts each of these strings as a list item into the provided DOM element. For example, `insertListItems(document.querySelector("#foo"), ["a", "b", "c"])` would insert three list items (i.e., `<li>` elements) into the DOM element with an ID of "foo".

```
function insertListItems(parent, list) {




}
```

8. JavaScript [15 Points]

A) (5 points) What is an anonymous function? What are two different ways to create an anonymous function in JavaScript?

B) (5 points) Write a JavaScript object literal named `todo` that contains a description of "Practice for the exam," a due date of "2021-10-12," and a priority of "high." Store the date as a Date object.

C) (5 points) Consider the following constructor function:

```
const Todo = function(description, dueDate, priority) {
    this.desc = description;
    this.due = dueDate;
    this.priority = priority;
};
```

Write a `toString` method that will apply to all `Todo` objects and returns a nicely formatted string of the todo item.

JavaScript Reference

These methods are correct and function as specified by the Mozilla JavaScript reference (`https://developer.mozilla.org/`). Parameters specified with `[]` are optional.

```
/*
 * The map() method creates a new array with the results of calling a provided
 * function on every element in this array.
 *
 * callback is invoked with three arguments: the value of the element, the index
 * of the element, and the Array object being traversed.
 *
 * If a thisArg parameter is provided to map, it will be passed to callback when
 * invoked, for use as its this value.
 */
arr.map(callback[, thisArg])

/*
 * The filter() method creates a new array with all elements that pass the test
 * implemented by the provided function.
 *
 * callback is invoked with three arguments: the value of the element, the index
 * of the element, and the Array object being traversed.
 *
 * If a thisArg parameter is provided to map, it will be passed to callback when
 * invoked, for use as its this value.
 */
arr.filter(callback[, thisArg])

/*
 * The reduce() method applies a function against an accumulator and each value
 * of the array (from left-to-right) to reduce it to a single value.
 *
 * reduce executes the callback function once for each element present in the
 * array, excluding holes in the array, receiving four arguments: previousValue,
 * currentValue, currentIndex, and array
 *
 * The first time the callback is called, previousValue and currentValue can be
 * one of two values. If initialValue is provided in the call to reduce, then
 * previousValue will be equal to initialValue and currentValue will be equal to
 * the first value in the array. If no initialValue was provided, then
 * previousValue will be equal to the first value in the array and currentValue
 * will be equal to the second.
 *
 * Note: If initialValue isn't provided, reduce will execute the callback
 * function starting at index 1, skipping the first index. If initialValue is
 * provided, it will start at index 0.
 */
arr.reduce(callback[, initialValue])

/*
 * The indexOf() method returns the first index at which a given element can be
 * found in the array, or -1 if it is not present.
 *
 * indexOf() compares searchElement to elements of the Array using strict
 * equality (the same method used by the === or triple-equals operator).
 */
arr.indexOf(searchElement[, fromIndex = 0])
```

This page is intentionally left blank.
You may use this page as working space.