CS 142 Midterm Examination

Spring Quarter 2023

You have 1.5 hours (90 minutes) for this examination; the number of points for each question indicates roughly how many minutes you should spend on that question. Make sure you print your name and sign the Honor Code below. During the examination you may consult two double-sided pages of notes; all other sources of information, including laptops, cell phones, etc. are prohibited.

I acknowledge and accept the Stanford University Honor Code. I have neither given nor received aid in answering the questions on this examination.

_____

(Signature)

_____

(Print your name, legibly!)

_____@stanford.edu

(Stanford email account for grading database key)

Only the front side of the exam pages will be scanned. Do not write answers on the back of the pages.

Problem #1 (12 points)

React is frequently used to build **single-page applications** that support **deep linking**. Answer the following questions, including explanations demonstrating your understanding of single-page applications, deep linking, and React.

A) Is it possible to build a single-page application without deep linking?

B) Would it be possible to support deep linking in an application that wasn't a single-page application?

… Problem #1 continued on the next page …

C) Would it be possible to construct a React application that wasn't a single-page application?

Problem #2 (6 points)

Both React and the JavaScript web application framework that preceded it (AngularJS) allowed the programmers to describe views using HTML-like code with templates containing JavaScript expressions embedded in it. Both React and AngularJS needed to detect if any of the template expressions changed to determine if it could display the view without having to run the code to re-render it. AngularJS did this by checking **all** the template expressions in a view to see if anything changed. This approach resulted in sluggish performance for views with many template expressions, such as a view containing a dense table of values.

Describe how React got around checking every JSX template expression when displaying these complex views that AngularJS had problems with.

Problem #3 (8 points)

Most web application frameworks, including the React framework we used in this class, follow the **model–view–controller** (MVC) software design pattern. Since React runs in the browser's environment that supports **HTML**, **CSS**, and **Javascript**, each of the MVC parts needs to be implemented using one of these three. For example, React controllers can use the JavaScript language, and its model data can be held in JavaScript data structures. Answer the following questions, including an explanation.

A) Can a React view also be written in JavaScript?

B) Can a React controller be written in HTML?

Problem #4 (12 points)

Assume you have the following HTML document:

```
<html id="H">
  <body id="B">
    <div id="D1">
      Div1
      <div id="D2">
        Div2
        <div id="D3">Div3</div>
      </div>
      <div id="D4">Div4</div>
    </div>
  </body>
</html>
```

If you looked up one of the `div` DOM nodes by ID (i.e., `getElementById` of `D1`-`D4`) you would be at the start of two singly-linked lists formed by the properties `offsetParent` and `parentNode`. The linked list formed by these properties always goes up the tree. We define the length of these lists as the number of DOM nodes on it (e.g., `e.offsetParent.offsetParent`) up to and including the `body` node. Note the `offsetParent` of the `body` node is null while its `parentNode` is the `html` tag. We don't include the `html` node in the length of either list.

A. Write a CSS style sheet for the HTML document that makes the `offsetParent` and `nodeParent` lists have the same length for every `div` node. If it is not possible, explain why not.

… Problem #4 continued on the next page …

B. Write a CSS style sheet that makes the `offsetParent` list longer than `parentNode` list. If it is not possible, explain why not.

C. Is it possible for children of a node (e.g. `D2` and `D4` of `D1`) to have different length lists? Answer the question with an explanation for both the `offsetParent` and `nodeParent` lists.

Problem #5 (8 points)

Assume you have the following HTML document (same as Problem #4):

```
<html id="H">
  <body id="B">
    <div id="D1">
      Div1
      <div id="D2">
        Div2
        <div id="D3">Div3</div>
      </div>
      <div id="D4">Div4</div>
    </div>
  </body>
</html>
```

Assume you have some JavaScript that adds both a bubble and a capture event 'click' handlers on every node that has an `id` property. The handler does a `console.log` of the event's phase (`'bubble'` or `'capture'`), `target.id`, and `currentTarget.id`.

A) If the user clicks on the string "Div2", show the `console.log` lines that will be written.

… Problem #5 continued on the next page …

B) If the user then runs the following JavaScript code and then clicks on the string "Div4", show the `console.log` lines that will be written.

```
function ev(event) {
    event.stopPropagation();
}
document.getElementById('D4').addEventListener('click', ev, false);
document.getElementById('D1').addEventListener('click', ev, true);
```

Problem #6 (6 points)

A) Explain why a new feature added to the JavaScript language standard can achieve wide usage much faster than additions to the language standards of compiled languages like Java or C++.

B) Consider the following code:

```
function MyObj() {
}
MyObj.prototype = {
  A: function A() { console.log("AP"); },
  B: function B() { console.log("BP"); },
  O: {C: function C() { console.log("CP"); }}
};
let o1 = new MyObj();
let o2 = new MyObj();

o1.A = function A() { console.log("AO"); };
o1.O.C = function C() { console.log("CO"); };
```

For each expression, state what the console statement would print or if an exception error would occur:

```
o1.A();


o1.B();


o1.O.C();


o2.A();


o2.B();


o2.O.C();
```

Problem #7 (8 points)

The introduction of CSS to HTML discourages the use of some of the original HTML tags as it was believed that CSS addressed the tag's functionality better. For each of the following HTML tags, briefly describe if they would be better done with CSS or their use in HTML is still encouraged.

A) `P`

B) `H1`

C) `PRE`

D) `IMG`

E) `B`

F) `DIV`

G) `TITLE`

Problem #8 (8 points)

An HTML document is served to a browser via the URL `http://localhost/a/b.html`.

What would be the URL sent to the web server if you clicked on the following hyperlinks in the HTML document?

A) `<a href="C.html">C</a>`

B) `<a href="/c.html">/c</a>`

C) `<a href="/c.html#c">/c#c</a>`

D) `<a href=":80#c">:80#c</a>`

Problem #9 (10 points)

When the following JavaScript code is executed, five log messages will be written to the console. Each log message will be two words.

```
try { console.log("one", varA); } catch(err) { console.log("one error"); }
function A() {
  try { console.log("two", varA); } catch(err) { console.log("two error"); }
  for (let i = 0; i < 1; i++) {
    var varA = 10;
    try { console.log("three", varA); } catch(err) { console.log("three error"); }
  }
  try { console.log("four", varA); } catch(err) { console.log("four error"); }
}
A();
try { console.log("five", varA); } catch(err) { console.log("five error"); }
```

A) Fill in the blanks to complete each log statement below. The first words have been given to you:

one _____

two _____

three _____

four _____

five _____

B) If we change the line "`var varA = 10;`" to be "**`let`** `varA = 10;`", what would the output be then?

one _____

two _____

three _____

four _____

five _____

Problem #10 (6 points)

Consider the following JavaScript function:

```
function A(arg) {
  if (arg)
    console.log("true", arg === true);
  else
    console.log("false", arg === false);
}
```

For each of the following invocations of the function A, list what the output will be:

A) `A(true);`

B) `A("false");`

C) `A(0);`

D) `A();`

E) `A([]);`

F) `A([0]);`

Problem #11 (6 points)

Closures in JavaScript can sometimes cause problems. One problem is large data structures can sometimes be captured in closures and cause the program to have unused memory that can not be collected by garbage collection. Write a function that captures something in a closure and returns something to the caller so that the caller holds the return value.

You can use a function call to `gigabyte()` to allocate and return a gigabyte-sized data structure.

```
function capture() {









}
```