# Sample CS 142 Midterm Examination

Spring Quarter 2018

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)

| Problem | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Score | | | | | | | | | | |
| Max | 10 | 10 | 9 | 12 | 12 | 8 | 10 | 9 | 10 | 90 |

## Problem #1 (10 points)

The following HTML document is loaded into a browser window:

```
<html>
<body>
  <div id="D">D
    <div id="A">A</div>
    <div id="B">B
      <div id="C">C</div>
    </div>
  </div>
  <script type="text/javascript">
     function callback(event) {
        console.log(event.currentTarget.id);
     }
     document.getElementById("D").addEventListener("click", callback, true);
     document.getElementById("A").addEventListener("click", callback, false);
     document.getElementById("B").addEventListener("click", callback, false);
     document.getElementById("C").addEventListener("click", callback, true);
  </script>
</body>
</html>
```

Hint: the last parameter to `addEventListener` is a boolean indicating capture phase (`true`) or bubble phase (`false`).

Part (a): Draw out what the browser would display for this document:

… problem continued from previous page…

Part (b): Write down what would be printed to the console log (order matters) if you clicked on the letters in alphabetical order.

Click A:

Click B:

Click C:

Click D:

Part (c): What changes, if any, would you need to make to the four event listeners to make the ids print as before but now always in alphabetical order? Please indicate your answer by filling in the blanks below.

```
document.getElementById("D").addEventListener("click", callback, _________);

document.getElementById("A").addEventListener("click", callback, _________);

document.getElementById("B").addEventListener("click", callback, _________);

document.getElementById("C").addEventListener("click", callback, _________);
```

Briefly explain your answer:

## Problem #2 (10 points)

```
<html>
<body>
<table>
  <tr class = "tr1">
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr class = "tr2" id = "tr1">
    <td class="td1">Alice</td>
    <td id = "td1"> 10</td>
  </tr>
  <tr class = "tr2">
    <td class="td2">Bob</td>
    <td>20</td>
  </tr>
  <tr class = "tr2" id = "tr2">
    <td id="td2">Carol</td>
    <td>30</td>
  </tr>
</table>
<br>
</body>
</html>
```

For each of the following CSS rule(s), write the words (e.g. “Alice”) that will appear red or blue in the browser and specify the color. If a word does not have a red or blue color you should not write it down. If no word changes color, you should just write down “None”.

Rules from one part do not apply to the next part. Briefly justify your answers for each part by stating what the CSS rules are doing.

A.

```
#tr1 {color: red;}
```

… continued from previous page..

B.

```
.td1 {color: red;}
```

C.

```
.tr2 .td1 {color: blue;}
```

D.

```
#tr2 .td1 {color: red;}
```

E.

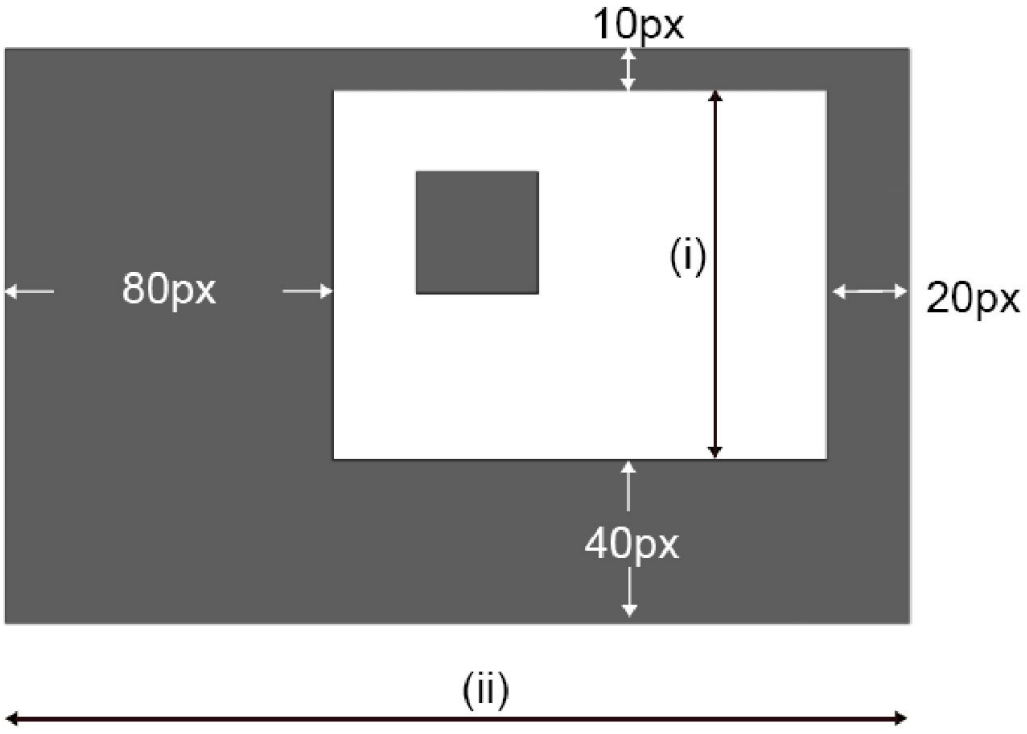
```
.tr2 {color: red;}
#td2 {color: blue;}
```

## Problem #3 (9 points)

```
<html>
<head>
      <style>
          div#outer{
            height:50px;
            width: 80px;
            color:white;
            border-color: gray;
            border-style: solid;
             border-left: _____________________________
             border-right: _____________________________
             border-top: _____________________________
             border-bottom: _____________________________
            padding: 20px;
             margin: 10px;
          }
          div#inner{
            background-color: gray;
            height: 30px;
            width: 30px;
          }
      </style>
</head>
<body>
      <div id = "outer">
        <div id = "inner"></div>
      </div>
</body>
</html>
```

The above HTML code could produce some boxes that look like the image on the following page (note that the image is not to scale).

… problem 4 continued from previous page.




A. Fill in the blank CSS rules above to create the border that has the weights indicated in the image.

B. Calculate the lengths of (i) and (ii). Write down brief explanations for your result.

## Problem #4 (12 points)

Given the following class and method definitions:

```
function Rectangle(length, width) {
      this.length = length;
      this.width = width;
}

Rectangle.prototype.hypotenuse = function () {
      var a_sq = Math.pow(this.length, 2);
      var b_sq = Math.pow(this.width, 2);
      var c_sq = a_sq + b_sq;
      return Math.sqrt(c_sq);
}

Rectangle.area = function() {
      return this.length * this.width;
}

var myVeryOwnRect = new Rectangle(5, 10);
```

Part (A): Please select which of the following will throw an error (there may be more than one). Briefly justify each selection.

```
   A. myVeryOwnRect.area();
   B. myVeryOwnRect.hypotenuse();
   C. Rectangle.area();
   D. Rectangle.hypotenuse(myVeryOwnRect);
```

… continued from previous page..

Part (B): It is not uncommon in JavaScript method functions to have a body whose first line is:

```
var self = this;
```

The remainder of the function body then uses `self` rather than `this` to access the object. Describe both the language feature of JavaScript and the usage that makes this a useful convention.

## Problem #5 (12 points)

A. Early single page web applications written in JavaScript did not support deep linking and rarely had '#' characters in the URLs they used. Explain why modern single page applications that support deep linking do have '#' characters in URLs.

B. Explain why deep linking was not really a problem before the advent of single page applications.

C. Describe what a browser does with an URL ending in #!/states when directed to a web page. Describe only the browser's functionality, not what some JavaScript might do with it.

## Problem #6 (8 points)

Consider the HTML snippet below:

```
<div id="mainDiv"><p>text 1a<span>text 1b</span></p><p>text 2a<span>text
2b</span></p></div>
```

A. Sketch out the DOM tree representation of this HTML snippet, showing only the parent/child relationships between the code. Non-text nodes in the tree should be labeled with their tag names, whereas text nodes should be labeled with their value. Hint: The "mainDiv" div is the root node of this DOM tree representation.

B. Assume you have access to a `mainDiv` object initialized with:
`var mainDiv = document.getElementById("mainDiv");`
Write a line of code that uses some combination of the DOM element properties `parentNode`, `nextSibling`, `previousSibling`, `firstChild`, `lastChild`, `innerHTML`, `textContent` to retrieve the string "text 2b" from the second span. You may not use any JavaScript string parsing functions or make any DOM API calls other than the ones listed above to do this.

## Problem #7 (10 points)

Consider the following AngularJS program shown in class:

```
<html ng-app>
  <head>
    <script src="./angular.min.js"></script>
  </head>
  <body>
   <div>
    <label>Name:</label>
    <input type="text" ng-model="yourName">
    <h1>Hello {{yourName}}!</h1>
   </div>
  </body>
</html>
```

It displays an input box that, when typed in, displays "Hello ", followed by the input. AngularJS also contains a directive `ng-if` that allows for conditional programming within the markup. Like its relative, `ng-repeat`, it creates a new scope inside the condition. If we add `ng-if="true"` to most of the lines the new scope has no effect on the program. For example if we change:

```
    <h1>Hello {{yourName}}!</h1>
```

to be:

```
    <h1 ng-if="true">Hello {{yourName}}!</h1>
```

the program continues to function correctly. Curiously, if we add it to the part of the template with the input tag:

```
    <input ng-if="true" type="text" ng-model="yourName">
```

the initial output looks the same but the characters typed into the input field are not displayed after the "Hello ". Describe what AngularJS is doing here to cause the modified program to work in some cases (e.g. `h1` tag) but not others (e.g. `input`) in this program.

## Problem #8 (9 points)

Part (A) In the Model-View-Controller Pattern, (1) describe what each of the Model, View, and Controller does and (2) how they appear in AngularJS:

A. Model

B. View

C. Controller

## Problem #9 (10 points)

AngularJS's two-way binding magically works most of the time yet AngularJS gives the programmer some control with `$watch` and `$digest` routines. It is possible to call $watch on a function (i.e. `$scope.$watch(watchFunction, …)` which will cause AngularJS to watch the value returned by the function. If we have a controller that installs a $watch on a function `watchFunction` and also invokes `$scope.$digest` frequently, which of the following can we say about the number of `$scope.$digest` calls made by the controller and the number of times `watchFunction` is called?

A. `watchFunction` will be called strictly less than the number of times `$scope.$digest` is called.
B. `watchFunction` will be called the same number of times that `$scope.$digest` is called.
C. `watchFunction` will be called strictly more than number of times that `$scope.$digest` is called.
D. There isn't a known relationship between `watchFunction` calls and calls to `$scope.$digest`.

Choose one and explain your answer.