CS 142 Midterm Examination Solutions

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) (Graded by Caitlin)

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:

This question explores what it means to be a markup language and what `div` does. The text will be displayed with each div starting a new line.
2 points (full credit)
D
A
B
C
Left justified, in one vertical line down the side of the browser

1 point
Showed some hierarchy of the div blocks (boxes)

… 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.

4pts total

Click A: **D, A**

Click B: **D, B**

Click C: **D, C, B**

Click D: **D**

Rubric: -1 for unique errors (if graders could tell consistent rationale)
Examples

- **Same error for click A and click B (AD/AB, DAAD/DBBD)**
- **Same error for click C and click D (assume propagates down: DBC/ DABC)**

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, False );

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

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

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

Briefly explain your answer:

All capture phase events are triggered (starting at outermost element) before all the bubble phase events (starting at the innermost element). Events are only triggered in the phase that they are registered! So to get D to run last no matter what, it must be in the bubble phase (2nd pass, most outer element). To get B to print before C, it must be in the capture phase. Since A is just a child of D, it doesn't matter what phase it's in (D always last). C is innermost, so it doesn't matter which phase it's in.

4 points total
Rubric:
3 pts, does your scheme work?

- **1.5 for B correct, 1.5 for D correct**

1pt, rationale

- **See above answer, demonstrated an understanding of the process**
- **Point given if the rationale coherently describes same misunderstanding in parts b and c and gives some show of event understanding**
    - **Ex. assumes that event propagates to all elements based on event listener type of element clicked (capture triggers all parents, bubble triggers all children)**
    - **Ex. capture phase events are played out before bubble phase events, but all events are triggered in capture and bubble phases**

Problem #2 (10 points) - Xiaoyan

```
<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;}
```

Alice,10 -> red
With reasonable justification (mention ‘#’ is id)

… continued from previous page..

B.

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

Alice -> red
With reasonable justification (mention '.' is class)

C.

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

Alice -> blue
With reasonable justification
(select elements that are in both class tr2 and td1)

D.

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

None
With reasonable justification
(No element has both id tr2 and class td1)

E.

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

Alice, 10, Bob, 20, 30 -> red
Carol -> blue
With reasonable justification
(#td2 is more specific than .tr2)

Rubrics:
2pts for each question from A to E
-1 pt for incorrect answer
-1 pt for wrong or no justification
Example: If there is no justification for all problems, -5pt in total.

- 0.5pt for correct answer but wrong color in A to D (minor error)
Example: C. Alice changes to red

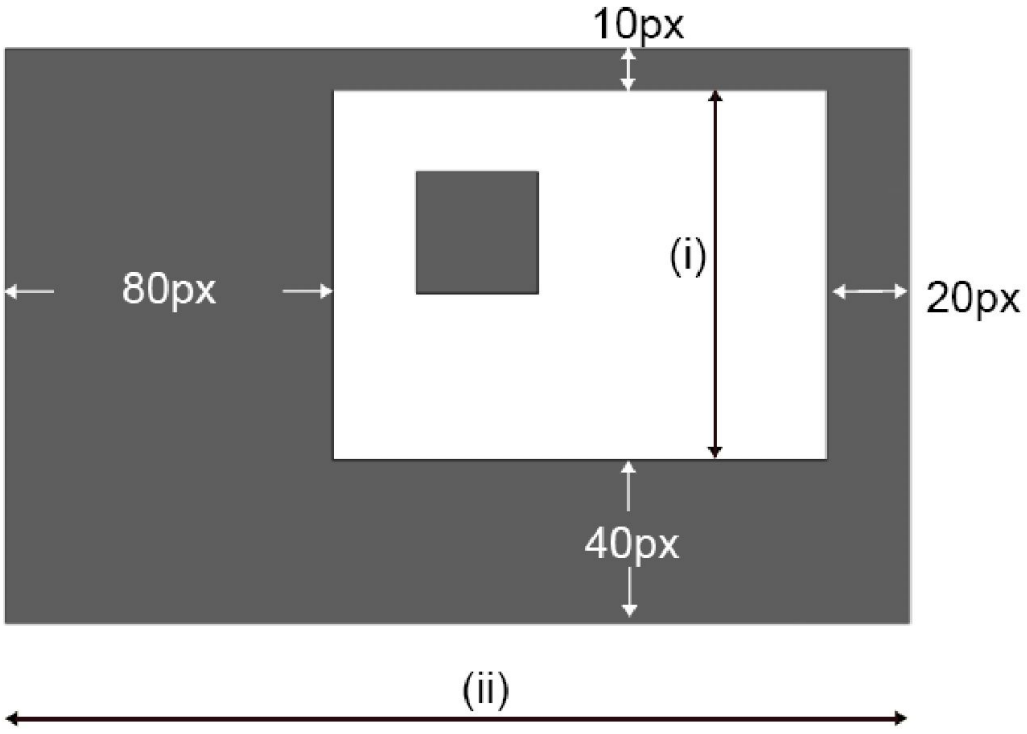
Common error: ".tr2 .td1" is only selecting the elements that are in both class tr2 and td1 for C and D, it is important to specify the "and" relation between the two class selectors.

Problem #3 (9 points) (Graded by Austin)

```
<html>
<head>
     <style>
         div#outer{
           height:50px;
           width: 80px;
           color:white;
           border-color: gray;
           border-style: solid;
           border-left: __________80px;__________
           border-right: _________20px;__________
           border-top: ___________10px;__________
           border-bottom: ________40px;__________
           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.

See blanks above. 1 point each. Common error: -2 for including margin in the border calculations (resulting in all answers being off by 10 px).

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

(i) length = (height of outer div) + (outer div top padding) + (outer div bottom padding)
= (50 px) + (20 px) + (20 px)
= 90 px

(ii) length = (width of outer div) + (outer div left padding) + (outer div right padding) + (outer div left border) + (outer div right border)
= (80 px) + (20 px) + (20 px) + (80 px) + (20 px)
= 220 px

2.5 points each. Common error: -3 for not including padding in calculations.

Problem #4 (12 points) - Andrew

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);
```

Two points awarded per correct answer (for a total of eight points). One point deducted for each incorrect explanation.

A will throw an error because `area` is defined as a static method on the Rectangle class. This would have only been valid if we defined it as `Rectangle.prototype.area`.

B will not throw an error, as the function was defined correctly.

C was ambiguous (depending on the environment one runs these commands in), so we were fairly lenient here. Points were awarded either way, though one point was taken off for incorrect explanations. If you were running this code on Node.js or in a browser with "use strict", `this` would be bound to undefined and trying to access the properties

`length` and `width` would throw an error. If the code was running in a browser without "use strict", `this` would be bound to the DOM window for which the properties `length` and `width` may or may not have values. Either way the function would return a value (likely NaN) but not strictly "throw an error".

D would throw an error because `hypotenuse` was not a static method. Many people put "doesn't take a parameter" here, but that wasn't (and wouldn't be) the cause of the error.

… 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.

We were looking for a reference to the fact that `this` changes depending on what function scope you are in. Full points were awarded if some allusion to ambiguous / changing `this` values, *and* if specific mention were made to functions within functions, event listeners, callbacks, etc. Some points were taken off if the intuition or direction was correct but the full explanation was not given.

Problem #5 (12 points) (Graded by Mendel)

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.

A single page application that doesn't support deep linking doesn't need to change its URL from the value it was started with. There wasn't any need for "#" character in the initial URL so they wouldn't have them. To support deep linking a SPA can record its state in the URL. By doing it in the fragment part of the URL the changes do not cause the browser to reset the running JavaScript environment.

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

Prior to single page applications web applications were built out of multiple pages each with unique URLs. With only a small amount of work, bookmarking and sharing of these URLs could be made to work as they did for any website. It was only when applications started running from a single page did deep linking require something special from the application.

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.

When browser encounters a fetch from an URL with a fragment, it will fetch the specified page if it is not already loaded and then look for an anchor tag or id that matches the fragment to scroll the window to.

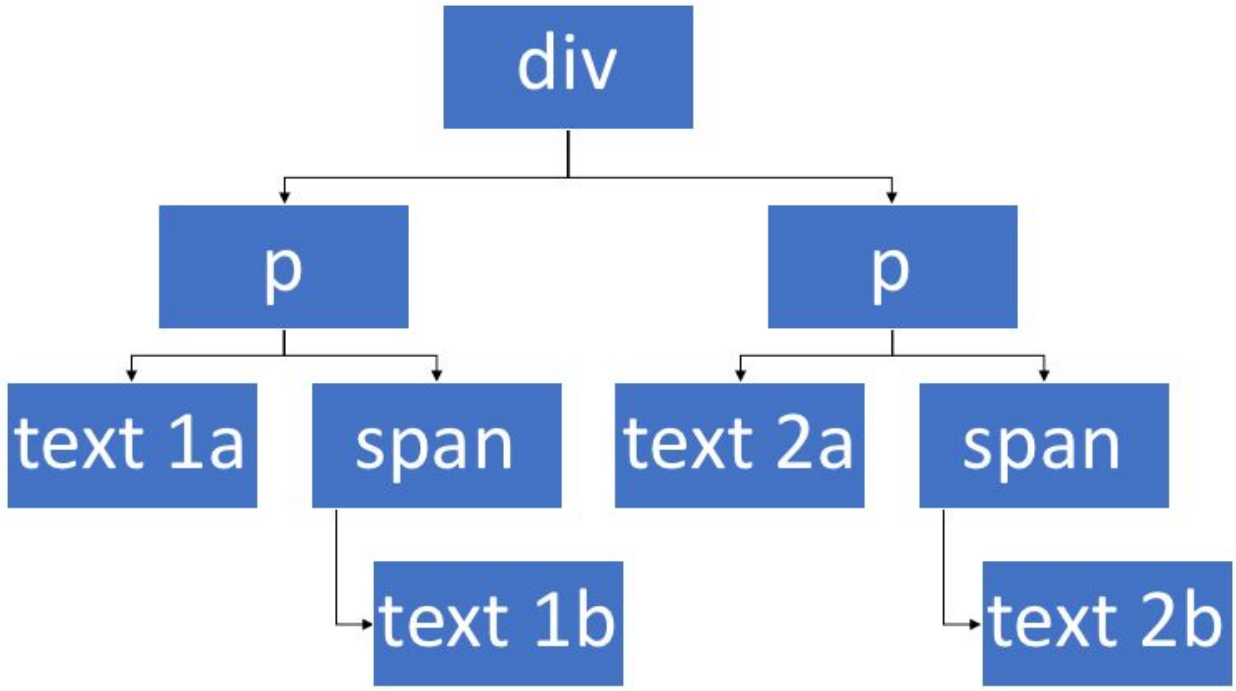
Problem #6 (8 points) (Graded by Don)

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.

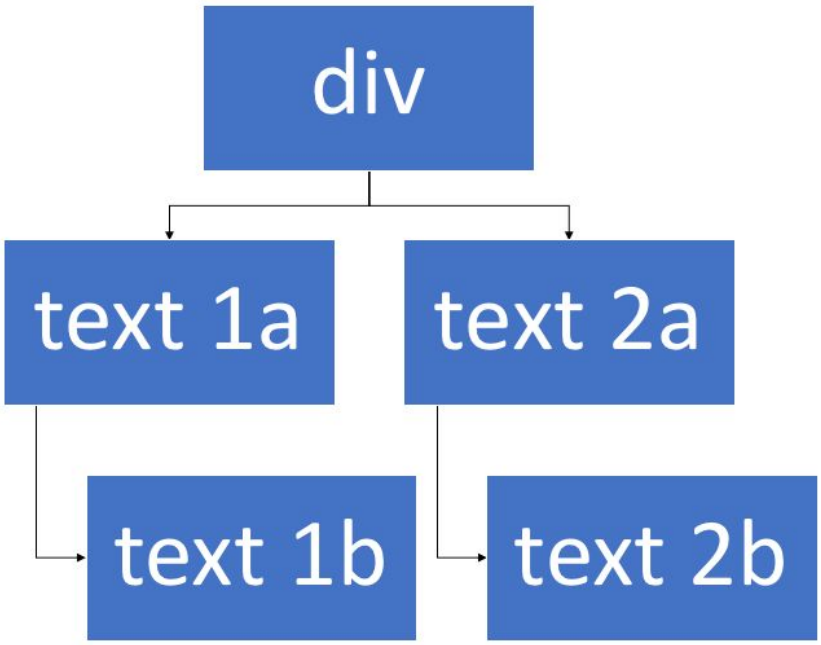
Correct solution:



Here, the arrows point from parent to child.

Grading rubric (max: -6):
-1 for each incorrectly placed arc
-1 for each missing arc
-1 for each extraneous arc
-1 for each incorrectly labeled node
-1 for each missing node
-1 for each extraneous node
-2 for creating separate nodes for opening and closing tags

Common mistakes:
Below is a common tree that people drew, which was unfortunately incorrect:



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.

Sample answers:
mainDiv.lastChild.lastChild.textContent
mainDiv.lastChild.lastChild.innerHTML
mainDiv.lastChild.lastChild.firstChild.textContent
Note: mainDiv.lastChild.lastChild.firstChild.innerHTML does not work (returns undefined).
Note: other answers worked (e.g., ones involving nextSibling) as long as you can extract the "text 2b" string.

Common mistakes:
mainDiv.lastChild.lastChild.firstChild
->This is incorrect since this gives you the object corresponding to the text node. The question asks you to retrieve the string "text 2b" from the second span.

Grading rubric (max: -2):
-1 for not selecting the right node
-1 for not obtaining a string from the innerHTML, textContent property
-2 for trying to access a property off the value obtained from reading textContent or innerHTML (e.g., innerHTML.textContent)

Problem #7 (10 points) - Ellen

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.

Answer:

- [3 points] The ng-if will create a new scope. Note that ng-if="true" means that the element it's attached to will always display.
- [2 points] For <h1>, the new scope is fine. The ng-if child scope's prototype points at its parent, so we just read off the yourName value from the parent.
- [5 points] For input, the yourName variable is written to, so it gets re-created in the child (input) scope, but the parent scope doesn't have access to the new yourName, so the parent never updates from its initial empty string value. So when the <h1> tries to read off yourName, it doesn't have the new value. This would be resolved by following the advice from class of using string properties of objects as ng-model arguments instead of just strings, so the variable isn't recreated. ("there should always be a dot in your model")

Rubric:

+3 for mentioning the new scope
+2 for explaining why the <h1> works (reads up the prototype chain to parent)
+2-5 for explaining the write issue with the <input>, depending on how much evidence was present that the student understood the scoping issue (e.g. something vague about primitive type edits not propagating upward would probably not earn the full 5)

Common errors:

- Students suggested that the input element wasn't displaying or that somehow the ng-if="true" condition could fail. (0 points)
- Students wrote "there should always be a dot in your model" without much justification (0-5 points depending on situation)

Problem #8 (9 points) - Hao

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

(1) stores and manages application data

(2) mention that model exsits as JavaScript objects on the angular front end

OR mention model often gets stored to Angular's scope variables/properties

OR mention ng-model and 2 way binding

(optional) mentions fetching from backend in JSON, XML, etc format

Common mistakes: JavaScript objects are not in JSON format, they can be converted to JSON objects

Model is not just a JavaScript file with a dictionary variable in it (this was just for the assignments, we faked the data this way)

B. View

(1) Defines what the applciation looks like

(2) Angular templates, directives, CSS (reasonable detail is enough)

Common mistakes:

for (2) just say html and css is not enough, these are not Angular specific

C. Controller

(1) mention fetches models and control view, handle user interactions, DOM even handling, or other key points

(2) mention several of the following: Appear as Angular controllers, tied to templates with ng-controller directive to form a component,

has angular services, handles events, etc. Or gave an example code snippet

Common mistakes:

Just saying controller in Angular is a file is not enough, we need more explanations here

Just saying controller is JS code is not enough, we need more explanations here

~~Problem #9 (10 points) (Graded~~ by Jeff)

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.

Answer (Graded by explanation, not only by letter choice):
During each digest cycle, all watched expressions are compared to their previously known value. Therefore, angular runs each watchFunction to check if the return value has changed. So, watchFunction would be called once per digest cycle. A call to $scope.$digest triggers a digest cycle, but digest cycles will also periodically be run by angular. Therefore, C is correct, watchFunction will be called strictly more than the number of times that $scope.$digest is called.

Half credit:
Correctly explaining that digest cycles call each watchFunction to check if the return value has changed. (However, these answers missed the fact that digest cycle is run automatically, or misunderstood some other part of the process).

No credit:
Misunderstanding the fact that digest cycles iterate through each watcher, or other major misunderstanding.

Common mistakes:
Claiming that watchers trigger digests.
Claiming that watchFunction will not be run if the value has not changed. However the function needs to be run to get the return value.