CS 142 Midterm Examination

Spring Quarter 2019

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)

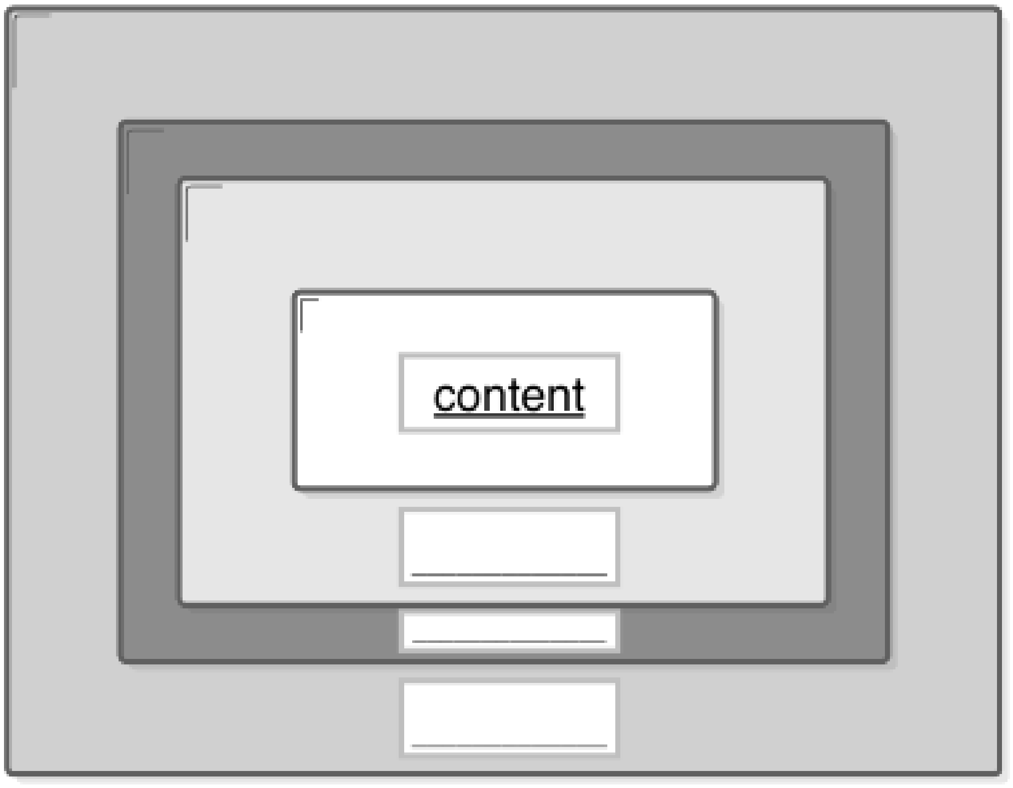
Solutions____
(Print your name, legibly!)

____@stanford.edu
(Stanford email account for grading database key)

Problem	#1	#2	#3	#4	#5	#6	#7	#8	#9	
Points	6	6	4	6	4	4	6	8	6	
Problem	#10	#11	#12	#13	#14	#15	#16	#17	#18	Total
Points	4	4	4	6	6	4	4	4	4	90

Problem #1 (6 points)

A. Fill in the blanks on the CSS box model diagram. Which is margin? padding? border?



Answer: Top to Bottom: padding, border, margin

B. What is the distance between the letters A and B if you had the following CSS rule?

```
A<div></div>B

div {
        height: 100px;
        width: 100px;
        padding: 10px;
        border-left-style: dotted;
        border-left-width: 5px;
        margin-top: 10px;
        margin-right: 20px;
        display: inline-block;
}
```

100 px (width) + 2 * 10 px (padding on either side) + 5 px (border-left width) + 20 px (margin-right) = 145 px

Problem #2 (6 points)

When discussing the idea called **Single Page Applications** (SPA) in class we introduced the term **deep linking**. Show your understanding of these two terms by answering the following questions about them.

A. The example single page applications we described in class supported deep linking. Could you have a single page application that didn't support deep linking? If so, describe what the shortcomings would be for it. If not, explain why not.

   You could have a single page application that didn't support deep linking. Without deep linking though, the URL wouldn't capture any context about the state of our app, and thus important modern features of apps such as refreshing, bookmarking and sharing links would be unsupported. Otherwise, the app would work fine. In fact, the early SPA were like this and lead to the term deep linking.

B. Could you have a web application that did not use the single page application approach (i.e. multiple pages) that supported deep linking? Justify your answer.

Yes, they support deep linking by default, by virtue of having multiple pages.

Problem #3 (4 points)

Most JavaScript frontend frameworks use the Model, View, Controller (MVC). Where Angular and VueJS spread the view and controller parts across different files, ReactJS combines the parts in a single file. For each of the blanks in the React component definition below, classify if the method function would be considered *view* or *controller*. Provide a brief explanation of your classification below your answer.

```
class Board extends React.Component {
  constructor(props) {
    super(props);
    this.state = { squares: [] };
  }
  componentDidMount() { Controller because we are fetching model information
    axios.get(`http://example.com/squares`)
      .then(res => {
        this.setState({ squares: res.data });
      })
  }
  handleClick(i) {      Controller because we are handling a user interaction
    const squares = this.state.squares.slice();
    squares[i] = 'X';
    this.setState({squares: squares});
  }
  renderSquare(i) {     View because we are primarily generating the html, in
other words what the application looks like
    return (
      <Square
        value={this.state.squares[i]}
        onClick={() => this.handleClick(i)}
      />
    );
  }
  render() { View because we are generating the html, in other words what the
application looks like
    return (
      <div>
        <div className="board-row">
          {this.renderSquare(0)}
          {this.renderSquare(1)}
          {this.renderSquare(2)}
        </div>
      </div>
    );
  }
}
```

Problem #4 (6 points)

An URL is composed of multiple components including:

A. Query parameters
B. Fragment
C. Port
D. Hostname
E. Scheme
F. Hierarchical portion

A. Identify by underline and labeling with the letters above (i.e. A-F) indicate where the above URL components are on this URL:

http://myth42.stanford.edu:3000/midterm/urls/links.html?year=3000&user=Mendel

```
____     ___________________     ____ ________________________    _____________________
E                 D                C              F                          A
```

B. Javascript Regular Expression literals can have many of the common punctuation characters in the character set. For example `/[.*+?^${}()|[\]\\#@%&;:]/` is a valid regular expression literal.
   1. Describe the issues that would arise if your web app needed to pass arbitrary regular expressions in URLs to the web app's backend.
   2. Describe how could you work around this problem.

1. Punctuation characters have special meaning in URLs, such as the "." or "-" character. Passing these as parameters could change the meaning of your URL and send your browser to an invalid/wrong URL. Many characters are also considered invalid (specifically, "unsafe" or "reserved") and may present security issues.

2. Use the URL encodings for your Regular Expression characters (%xx) instead of the character literals

Problem #5 (4 points)

Your browser currently is on `http://cs.stan.edu/a/b/c.html`. Write the resulting URLs for clicking on each of the following links:

A. `<a href=”123.html”>a</a>`

`http://cs.stan.edu/a/b/123.html`

Since the URL starts with only hierarchical part and doesn’t start with a ‘/’ char, this is a relative URL. The new URL will keep all but the last component.

B. `<a href=”/234.html”>b</a>`

`http://cs.stan.edu/234.html`

The URL starts with ‘/’ so it is an absolute path. The new URL’s hierarchical component will be completely replaced.

C. `<a href=”#c”>c</a>`

`http://cs.stan.edu/a/b/c.html#c`

The ‘#’ character indicates we are navigating to a fragment on the current page. So, the URL remains the same, with the fragment added on to the end.

D. `<a href=”http://google.com”</a>`

`http://google.com`

An entirely new URL is specified, so we navigate away from our current page. Credit was also given to those who noticed the missing ‘>’ character on the opening tag.

Problem #6 (4 points)

```
class MyComponent extends React.Component {
    constructor(props) {
      super(props);
      this.state = {
           pageStatus: 'rendering...',
      };
    }

    updatePageStatus() {
      this.setState({
          pageStatus: 'rendered!',
      });
    }

    render() {
      this.updatePageStatus();
      return (
          <div>
            {this.state.pageStatus}
          </div>
      );
    }
}
```

If we added this component to the view, our app would crash. Describe what the problem is here.

the problem is that this.setState() triggers a render() call in react, which causes an infinite loop between render() and this.updatePageStatus()

Problem #7 (6 points)

You are given the following HTML document:

```
<html>
  <head>
  </head>
  <body>
    <div id="container"></div>
  </body>
</html>
```

Write JavaScript code using the DOM to make the document look like this:

```
<html>
  <head>
  </head>
  <body>
    <div id="container">
      <p class="pClass">My paragraph.</p>
      <span id="mySpan" style="visibility: hidden;">My span.</span>
    </div>
  </body>
</html>
```

You may NOT use assignment to the innerHTML in your solution.

Example solution:

```
var container = document.getElementById('container');

var newP = document.createElement('p');
newP.className = 'pClass';
newP.textContent = 'My paragraph.';
container.appendChild(newP);

var newSpan = document.createElement('span');
newSpan.id = 'mySpan';
newSpan.style.visibility = 'hidden';
newSpan.textContent = 'My span.';
container.appendChild(newSpan);
```

2 points awarded for correctly accessing the container, 2 points for correctly creating/adding <p>, 2 points for correctly creating/adding <span>

Problem #8 (8 points)

Given the following HTML file, what will the five DOM expressions below return?

```
<html>
  <head>
    <title>getElementById example</title>
  </head>
  <body>
    <div id="div1">CS142 div1</div>

    <div id="div2">
      <div id="div3" class="coolDiv">CS142 div3</div>
      <span id="div4" class="boldText">CS142 div4</span>
      <div id="div5" class="redDiv">CS142 div5</div>
    </div>

    <div id="div6">
      <h2>This is a header.</h2>
      <p>This is a paragraph.</p>
      <b class="boldText">This is bold.</b>
      <i>This is italic.</i>
    </div>
  </body>
</html>
```

A. `document.getElementById('div1').textContent` CS142 div1

B. `document.getElementsByTagName('p')[0].innerHTML` This is a paragraph.
The <p> tags are not included.

C. `document.getElementsByClassName('boldText').length` 2
We search the entire document and there are 2 elements wil class="boldText"

D. `document.getElementById('div5').parentNode.nextElementSibling.firstElementChild.tagName` H2

E. `document.body.firstElementChild.nextElementSibling.nextElementSibling.getElementsByTagName('span').length` 0
We only search div6, which has 0 <span> elements

Problem #9 (6 points)

You are initially given this simple HTML document containing two buttons:

```
<html>
  <head>
  </head>
  <body>
    <button id="cs109Button">Enroll in CS 109</button>
    <button id="cs142Button">Enroll in CS 142</button>
  </body>
  <script>
      function Course(courseName) {
        this.courseName = courseName;
        this.id = courseName + '_id';

        document.getElementById(courseName + 'Button').onclick = function(e){
           console.log('The target for this event is  ' + e.target.id);
           console.log('The currentTarget for this event is  ' +
                                                      e.currentTarget.id);
           console.log('Successfully enrolled in ' + this.courseName);
         };
       }

       var cs142 = new Course('cs142');
  </script>
</html>
```

A. When you click on the button with the text "Enroll in CS 142", what gets printed to the console? Explain why.

The target for this event is cs142Button
The currentTarget for this event is cs142Button
Successfully enrolled in null/undefined
The target and currentTarget ids are both cs142Button because the it is the element that triggered the event (target) as well as the element the event is registered on (currentTarget). this.courseName is undefined because in the scope of the event listener, this refers to the element the listener is being registered on, ie the button element.

B. When you click on the button with text "Enroll in CS 109", what gets printed to the console? Explain why. Nothing because we did not initialize a Course object for 'cs109', which would have installed an event listener on that button.

Problem #10 (4 points)

You are given the following HTML document and JavaScript which adds event listeners to some of your document elements (hint: the 3rd argument of addEventListener is useCapture):

```
<html>
  <head>
  </head>
  <body>
      <div id="outerDiv">
            div id="innerDiv">CLICK ME</div>
      </div>
  </body>
  <script>
      var body = document.body;
      var innerDiv = document.getElementById('innerDiv');
      var outerDiv = document.getElementById('outerDiv');

      innerDiv.addEventListener("click", function (e) {
        console.log('div-inner-bubble');
      }, false);

      outerDiv.addEventListener("click", function (e) {
        console.log('div-outer-bubble');
      }, false);

      body.addEventListener("click", function (e) {
        console.log('body-bubble');
      }, false);

      body.addEventListener("click", function (e) {
        console.log('body-capture');
        e.stopPropagation();
      }, true)
  </script>
</html>
```

What gets printed to the console when you click the words "CLICK ME"? Explain your answer:

‘body-capture’ is the only thing printed out. The capture phase comes first, and so the fourth event listener is triggered first. It prints out and then it stops propagation for the event. This means that the event will not propagate to the event listeners that would be triggered later.

Problem #11 (4 points)

If this program is executed, what will be logged on the console?

```
function Friend() {
  this.arrive = function () {
    console.log('Hello');
  };
}

var friend1 = new Friend();
var friend2 = new Friend();
friend2.arrive = function () {
  console.log('Hi');
};
friend1.arrive();
friend2.arrive();

Friend.prototype.leave = function () {
  console.log('Bye');
};
friend1.leave();
friend2.leave();
var friend3 = new Friend();
friend3.leave();
friend3.leave = function () {
  console.log('Adieu');
};
friend3.leave();
friend2.leave();
```

```
Hello
Hi
Bye
Bye
Bye
Adieu
Bye
```

Problem #12 (4 points)

For each of the testing programs list below, classify them as examples of:

- unit testing
- end-to-end testing

Briefly explain your answer.

A. A program opens a browser, navigates to a web app, and makes sure that the log in and sign in process works.

End to end testing. This program is testing various parts of the application, as opposed to a single component. It also programmatically interacts with actual browser / application.

B. A program checks that a sidebar component collapses correctly.

Unit testing: This program tests a very specific function of a single component (ie. just the collapsing property of the sidebar).

C. A program utilizes mock components and DOM for testing.
Unit testing. This program mocks components and the DOM as opposed to interacting with the actual application, which is a feature of unit testing.

D. A program containing:

```
describe('sorting the list of users', function() {
   it('sorts in descending order by default', function() {
      var users = ['jack', 'igor', 'jeff'];
      var sorted = sortUsers(users);
      expect(sorted).toEqual(['jeff', 'jack', 'igor']);
   });
 });
```

Unit Testing. This code block just tests a single function (sort), independently of the web application.

Problem #13 (6 points)

JavaScript contains two keywords for declaring a variable: `var` and `let`. Although both declare variables without specific types, they behave differently. For each of the following two code fragments, state what the console log output would be if `VARLET` was set to `var` and `let`. Assume that code is run using "use strict" JavaScript rules.

Code Fragment A	**Code Fragment B**

`try {`
`  i = 10;`
`  for (VARLET i = 0; i < 5; i++) {`
`     console.log(i);`
`  }`
`  console.log(i);`
`} catch (err) {`
`  console.log("ERROR");`
`}`
	`try {`
	`  for (VARLET j = 0; j < 5; j++) {`
	`     console.log(j);`
	`  }`
	`  console.log(j);`
	`} catch (err) {`
	`  console.log("ERROR");`
	`}`

A: VARLET = **`var`** `console log output:`	**B:** VARLET = **`var`** `console log output:`
0 1 2 3 4 5	0 1 2 3 4 5
A: VARLET = **`let`** `console log output:`	**B:** VARLET = **`let`** `console log output:`
ERROR	0 1 2 3 4 ERROR

Problem #14 (6 points)

The functions and properties defined in JavaScript class definitions end up being stored in various objects when instances of the class are created. Some of the objects used include:

- Instance - The instance object
- Prototype - The prototype object of an instance
- Constructor - The constructor function of the class

For each of the parts of the class listed below, state which of the above objects they end up in. Provide a brief explanation of your answer.

A. Class methods

Our intended answer was prototype. However, we realized there was some ambiguity with the phrase class methods so we also accepted constructor. When you create a method for a class you want to add it to the prototype so that you can be efficient with memory. You would therefore do Classname.prototype.methodName = function ()...

B. Class static properties

Class static properties are used when you want to have information associated with a particular class but it doesn't rely on any specific information about an instance. If I had function Car and every car made the same sound, I could do something like Car.sound = "Honk"; note that to use this, I would write Car.honk. I could not do Car redCar = new Car(); redCar.honk; Because functions even constructors are treated like objects, you can add properties to them. The answer is therefore constructor.

C. Instance properties

An instance property is the information that distinguishes different objects of the same class. For instance, if I had class Rectangle would have instance properties associated with width and height. This needs to be in the instance of the object because it is unique to the object.

Problem #15 (4 points)

Write a JavaScript function (create_a_closure) that causes a closure to be created when called. The created closure should contain the variables `myNum`, `myStr`, `myBool` declared below:

```
let myNum = 1;
let myStr = '1';
let myBool = true;

function create_a_closure() {

      // Closures are create with function definitions
      function makeAclosure() {
            // To make a closure contain a variable we just need to access it
            return [myNum, myStr, myBool];
      }
     return makeAclosure;

}
```

Common mistakes:

- Redeclaring myNum, myStr, and myBool inside of create_a_closure. This definitions hide the global variables and cause them not to be in the closure.
- Calling makeAclosure (e.g. `makeAclosure()`). This would result in a closure being created and then immediately destroyed. Since this seemed to be compatible with the problem statement (create a closure) it was acceptable.

Problem #16 (4 points)

For each of the JavaScript code fragments below, describe what would be printed to the console log when the code is run. Recall the `setTimeout` function take an arguments a function and number of milliseconds and will call the function in the specified number of milliseconds.

A.
```
for (var  x = 1; x < 5; x++) {
   setTimeout(function () { console.log(x); }, 1000*x);
}
```

The above loop contains a function definition that will create a closure pointing at the variable 'x'. Four functions with closure contain x well be created and scheduled to run in 1, 2, 3, and 4 seconds. When they run x will have been set to 5 (the exit contain of the loop). So the output will be:

5
5
5
5

B.
```
function f(a) { console.log(a); }
for (var  x = 1; x < 5; x++) {
   setTimeout(function () { f(x); }, 1000*x);
}
```

As above, the loop will create four functions with closures pointing at x. Also as above, they won't run into the loop has finished. At that time x will be 5. When the functions run they call function f with an argument of 5 (f(5)) so the output will be:

5
5
5
5

Problem #17 (4 points)

```
class ClickComponent extends React.Component {
    constructor(props) {
      super(props);
      this.handleClick1 = this.handleClick1.bind(this);
    }
    handleClick1() {
       console.log("Handle Click 1 called");
    }
    handleClick2() {
       console.log("Handle Click 2 called");
    }

    render() {
      return (
        <div>
          <div onClick={this.handleClick1}>Click #1 using .bind</div>
          <div onClick={() => this.handleClick2()}>
             Click #2 using arrows
          </div>
          <div onClick={() => this.handleClick1()}>
            Click #3 using combined
          </div>
        </div>
      );
    }
}
```

The event handling patterns use on the <div> regions with content "Click #1 ..." and "Click #2 ..." follow one of the standard React patterns for handling events discussed in lecture. The "Click #3 ..." uses a strange combination of both the .bind and arrow function approaches. Would this work? Justify your answer.

This will still work they way we want it to. Because we use the arrow function, the context of this does not change and still refers to ClickComponent. And (while redundant), in the constructor we bound the function to the ClickComponent.

Problem #18 (4 points)

Use the definition of `ClickComponent` from Problem #17 in the question.

If you created one of these ClickComponent (e.g. had `<ClickComponent />` in some render function) somewhere inside of the ReactJS runtime it would do a

```
    new ClickComponent()
```

which would create a new object of class ClickComponent. Like all class JavaScript objects, this newly created object (**`object`**) would have a prototype object (**`prototype`**). Fill in the table below indicating if the specified code fragment would generate accesses to these two objects (**object** and **prototype)** by:

Marking R if the code fragment reads the object.
Marking W if the code fragment writes (or reads and writes) the object.

Leave the cell blank if no access is made. Note for the onClick fragments assume that click event happens.

JavaScript Code Fragment	**`object`**	**`prototype`**
`this.handleClick1 = this.handleClick1.bind(this);`	**W**	**R**
`onClick={this.handleClick1}`	**R**	
`onClick={() -> this.handleClick2}`	**R**	**R**
`onClick={() -> this.handleClick1}`	**R**	

The first row has an assignment to `this.handleClick1` which will generate a write to the object. The write will create the property `handleClick1` in the object. The right hand side of the assignment does a read of `handleClick1` first in the object and since it is not there (yet) it will go up to the prototype object and read it to find the handleClick1 routine.

The second row accesses and finds the newly created `handleClick1` in the object so is a read of the object. No going up the prototype chain is needed.

The third row does a read of handleClick2. It will first check the object and not finding it will go up and look in the prototype.

The fourth row does a read of handleClick1. It will first check the object and finds it.

Additional page to make page count even - no problem.