Question 1. [5 marks]

Indicate whether each statement is True or False by circling the appropriate answer.

TRUE FALSE The Internet Protocol uses the three-way handshake to maintain a connection between two hosts.

TRUE FALSE `'use strict';` will sometimes (but not always) cause the following line of code to fail:
`a = 7;`

TRUE FALSE Object constructor functions are of the same type in JavaScript as other functions.

TRUE FALSE Anonymous functions must be called immediately where they are defined (i.e., immediately invoked).

TRUE FALSE The Transport layer is responsible for managing packets as they travel between network nodes.

Question 2. [4 marks]

The following three lines of JavaScript are run, creating three variables in the global scope:

```
let a = 4;
let b = 1;
var s = { a: 1, b: 4, c: 2 };
```

The code fragments below are each run directly after the above three lines. They are run **independently** of each other. Beside each code fragment in the table below, write the console output when the code fragment is executed after the above three lines. If the code would cause an error, write ERROR and give a brief explanation.

Code	Output or Cause of Error

`(function () {`
`  if (a > 2) { let b = 3; a = 0; }`
`  console.log(a + b);`
`})();`	

`var c = 5;`
`for (const i = 0; i < 3; i++) {`
`  if (c < 4) {`
`  console.log(c);`
`  }`
`  var c = 3;`
`}`	

`(function(o) {`
`  o.a = 4;`
`  var s = o;`
`  s.b = 1;`
`})(s);`
`console.log(s.a); console.log(s.b);`	

`function foo() {`
`  a(a);`
`  function a(b) {`
`  a = 7; b = 2;`
`  }`
`  console.log(a);`
`}`
`foo(); console.log(a);`	

Question 3. [5 marks]

Consider the code below. Fill in the boxes such that the output of the `console.log` statements at the end of the code is as specified by the comments beside them.
The boxes cannot contain object literals (i.e., objects surrounded by curly brackets `{}`).

```
const Fruit = {
    constructor: function(name, season) {
        this.name = name;
        this.season = season;
    }
}

function Apple(p) {
    [          ].bind([          ])([          ]);
}

function fruits() {
    this.type = 'Fruit'
}

Apple.prototype = Object.create([          ]);

const obj = new [          ]({ name: 'strawberry', season: 'summer' });

console.log(obj.name);  // 'strawberry'
console.log(obj.season);  // 'summer'
console.log(obj.type);  // 'Fruit'
```

Question 4. [10 marks]

In this question, you will implement a small web page based on some user requirements. You will write some HTML and CSS, a JavaScript library, and a JavaScript file that will use the library you wrote to perform the requirements of the site.
The following are the requirements for the web page:

- The page has two buttons: `Add Square` and `Add Circle`.
- The two buttons will add square or circle shapes to the DOM when clicked.
- Squares should be blue and 15 by 15 pixels, and should have the text `SQUARE` somewhere inside of them. Circles should green, with a diameter of 10 pixels, and should have the text `CIRCLE` somewhere inside of them.
- All shapes should be 10 pixels away from other shapes in any direction.
- Any time you add a square, it should be added on a new line (anywhere under the most recently added shape).
- Circles added after a square should be added on a new line (anywhere under the square).
  Any time you add two or more successive circles, each one should be added right beside (to the right of) the most recently added circle.
- The shapes should not be added to the `<body>` element, and should not be put in the same HTML element as the buttons.

Part (a) [2 marks]

Complete the HTML body below for the above requirements. It should have all elements necessary to start adding shapes, as well as **one square** and **one circle** already on the page.
You should provide proper `class`es and/or `id`s as needed. You will use these in the next parts to style and create user interactions. **Do not** write **any** JavaScript in this box.

```
<!DOCTYPE html><html>
    <head><link rel="stylesheet" type="text/css" href="shapes.css"></head>
    <body>

    <div id='buttons'>




    </div>







     <script type="text/javascript" src="ShapeMaker.js"></script>
     <script type="text/javascript" src="shapes.js"></script>
    </body>
</html>
```

Part (b) [3 marks]

Complete the file `shapes.css` to define the styling for the shapes per the requirements. Make sure the selectors match what you put in the HTML.

```
shapes.css
-----------------------------------
```

Continued..

Part (c) [3 marks]

Complete the JavaScript library `ShapeMaker.js` below. It should have functionality for adding squares and circles to the proper place in the DOM. Make sure you are considering your HTML and CSS files when writing your code.
All DOM elements and text nodes must be created dynamically. You may not use .innerHTML, .innerText, or anything similar. Doing so will result in a 0 for this part.

```
ShapeMaker.js
---------------------------------------
function ShapeMaker () {
    // empty function body.
}

ShapeMaker.prototype = {
        addSquare: function() {









        },

        addCircle: function() {









        }
}
```

Continued..

Part (d) [2 marks]

Now, write the code for `shapes.js`, which should set up the user interactions for the buttons when they are clicked. You should use the `ShapeMaker` library and call the appropriate functions at the appropriate times. All interactions should be set up in this file - no JavaScript should be in your HTML.

```
shapes.js
----------------------------------
```

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

Last Name: _____ **First Name:** _____

HTML and CSS

```
<body>, <div>, <h1>, <h2>, <p>, <ul>, <ol>, <li>,
<span>, <strong>, <em>
<img src=''>
<form>
<input type=''>
<button>

height, width, position, display, padding,
margin, border, border-radius,
color, background-color, font-family, font-size
```

JavaScript DOM functions/methods

```
document.getElementById(id)
document.getElementsByClassName(class)
document.querySelector(selector)
document.querySelectorAll(selector)

document.createElement(string)
document.createTextNode(text)
element.appendChild(element)

element.setAttribute(attributeName, value)

element.addEventListener(event, function)
event.preventDefault()

Properties:
element.className
element.id
element.classList
element.value
element.parent
element.children

array.push(object)
parseInt(string)
```