1 How to use this review guide

The problems contained within this document are representative of what will appear on your final exam for this class. This does not imply that only the problem types that appear below are included on your exam. For instance, we might not directly reference the `Dice` activity, but you are expected to be able to implement pieces of that logic. The bulk of this document presents programming problems, but your exam in not comprised entirely of questions that require that you write lots of JavaScript. You should expect questions about CSS and HTML as well.

1. Use HTML to format the following. (To conserve space, we did not put in the customary vertical spacing for these first questions; write them on a separate sheet of paper if you need to do so.)

   (a) Use the `<br>` element to reproduce the following text:

   ```
   This is the
   middle --the true middle--
   part of this sentence.
   ```

   (b) Use the `<pre>` element to format the following code fragment:

   ```
   if( x < 0 ) {
       alert(x + " is negative!");
   } else {
       alert(x + " is okay.");
   }
   ```

   (c) Use HTML to construct the paragraph text and the underlined hyperlink below:

   This is my Homepage.

   (d) Use HTML to construct the paragraph text and the underlined mail link below:

   This is my Email address.

2. Examine the following code fragment:

```
var str="";
for( var index = 0; index <= 10; index++ )  {
   if( index % 4 === 0 ) {
      str = str + index + " ";;
   }
}
window.alert( str );
```

   What will be the value of `str` as it appears in the alert box when this code is executed?

2. ____________

3. Assume that the procedure `selectSort` works as we have developed it in class. Given

```
var myArray = [ 3, 0, 2, 1  ];
```

   (a) In the space below, re-write `myArray` as it would appear after *one* iteration of the `selectSort` procedure:

   (b) In the space below, re-write `myArray` as it would appear after *two* iterations of the `selectSort` procedure:

4. This question contains two parts: In part (a), you will write a function that returns `true` if a `word` (a string) is "common," i.e., that it contains *any* of the letters from the `commonLetters` variable. In part (b), you will use this function to count the number of words (`string`s) in an array of `strings` that are "common" as determined by the function in part (a). Assume that all strings are in lowercase!

   (a) Write the `isCommon()` function that returns `true` if the `word` (a `String`) contains *any* of the letters in the `commonLetters` variable.

   ```
   var commonLetters ="etaoin";
   function isCommon( word ) { // start here
   ```

   (b) Assume that whatever you wrote for part (a) is correct, and use that function to implement `countCommonWords( strArray )` that takes an array of `string`s and returns the number of these that are "common words."

   ```
   function countCommonWords( strArray ) {
      var count = 0;
      // start here.
   ```

5. Complete the partially-written function below that takes `target`, an integer, and `intsArray`, an array of integers, and returns the *rightmost* index of the `target` or `-1` if `target` is not found in `intsArray`. The programmer's plan is to examine the `intsArray` from right to left and return the first occurrence of `target`, or `-1` if it is not found.

```
function indexOf( target, intsArray ) {
    var start = <expression-1>; // part(a)

    while( start <operator-1> 0 ) { // part(b)
       if( intsArray[ start ] <operator-2> target ) {
          return start;
       }
       start = start -1;
    }
    return -1;
}
```

(a) Write the correct JavaScript for `<expression-1>` in the function above:

(b) Write the correct JavaScript operator for `<operator-1>` in the function above:

(b) ______________

(c) Write the correct JavaScript operator the `<operator-2>` in the functino above:

(c) ______________

2 Additional study suggestions

Carefully review the last two classroom activities —especially the logic and mechanism for generating dynamic data, such as the rows of tables. Make sure that you understand the relationships between the code and the HTML elements, in particular the use of the `id` attribute.

Although we do not require that you write object definitions for the exam, we do expect you to be able to use the methods and access the properties that appear in the table (reference) at the end of this document.

- Understand the difference between the `indexOf( char )` and `charAt( index )` methods as they are defined and used on the `string` class.
- Understand how the remainder operator, which is written in JavaScript as `%`, is used. If you are unsure, write a small function that uses this operator:

  ```
  function testRemainder( range, divisor ) {
     var answer = "";
     for( var number=0; number <= range; number++ ) {
         answer += "number + "%" + divisor + " is " ( number % divisor ) + "\n";
      }
      window.alert( answer );
  }
  ```

  Place this in an HTML file that contains some button to activate it: maybe something like:

  ```
  <button onclick="testRemainder( 15, 4 );">Click to see remainders</button>
  ```

- Be very careful reading `for` statements. Don't assume the obvious:

  ```
  for( var index=0; index < someArray.length; index++ ) { ... }
  ```

  is *very different* from (assuming that `someArray` contains more than 1 element):

  ```
  for( var index = someArray.length - 1; index >= 0; index-- ) { ... }
  ```

Additional Functions allowed on this exam

Dice Object Properties and Methods

Use only the following properties and methods that we defined for the `Dice` object in class:

Signature	**Description**
`new Dice( numFaces )`	Creates a *new* `Dice` object with the corresponding number of faces. Usage: `var myDice = new Dice( 6 )`.
`numFaces`	Returns a non-negative integer that gives the number of faces belonging to the `Dice` object. Usage: `myDice.numFaces` (which returns 6 from the example above).
`roll()`	Updates the `face` of a `Dice` object with a value in the range of 1 through `numFaces` (inclusive). Usage `myDice.roll()`.
`getCount()`	Returns a non-negative integer indicating how many times this `Dice` object has been rolled. Usage: `myDice.getCount()`.
`face`	Returns the current face shown on this `Dice`. Usage `myDice.face`, which returns an integer between 1 and six, based upon the example used in this table.

String Object Properties and Methods

Use only the following properties and methods that we used for the `String` object in class:

Signature	**Description**
`length`	Returns a non-negative integer that gives the length of the string. Usage: `myString.length`. Usage: if `myString="Tom"`, then `myString.length` returns 3.
`charAt( index )`	Returns the character at the `index` in the string. Note, if `index` is out of range, an error is signaled. Usage: If `myString="Tom"`, then `myString.charAt(0)` returns "T."
`indexOf( char )`	Returns a non-negative integer that gives the first location of the `char` in the string scanning from left to right. If `char` is not found in string, then a negative integer is returned. Usage: `myString.indexOf("T")` returns 0, using the example from above. But, `myString.indexOf("a")` returns -1.

Sketch of the Selection Sort Algorithm

```
procedure SELECTSORT(array)
    for $index = 0$; $index < array.length$; $index \leftarrow index + 1$ do
        $minIndex \leftarrow$ INDEXOFMIN($index, array$)
        SWAP($index, minIndex, array$)
    end for
end procedure
```