Cohort Exercise

Total 10 Marks

You may submit your codes in a zip file with folders and subfolders, and a report with explanation.

Task 1 (2 Marks)

Complete the following HTML and JavaScript codes, so that when the button `Submit` is clicked, the min and the max of the sequence of numbers (seperated by ",") entered in the input text box will be displayed in the `span` elements.

```
<!DOCTYPE html>
<html>
    <head>
        <script src="minmax.js"> </script>
        <meta charset="utf-8"/>
        <title>50.003 sample code: Min and Max </title>
    </head>
    <body>
        <div>Your input: <input id="textbox1" type="text" /> </div>
        <div>Min: <span id="min"></span></div>
        <div>Max: <span id="max"></span></div>
        <div><button id="button1">Submit</button></div>
    </body>
</html>
```

```
function numbers(l) {
    var o = [];
    for (let i in l) {
        var n = parseInt(l[i],10);
        if (!isNaN(n)) {
            o.push(n);
        }
    }
    return o;
}
// input: an array of numbers
// output: an object containing 'min', with the minimum of the array
//           and 'max' the maximum of the array.
function min_max(a) {
    var min = null;
    var max = null;
    // TODO: fixme
    return { 'min' : min, 'max' : max}
}

function handleButton1Click() {
    var textbox1 = document.getElementById("textbox1");
    var min = document.getElementById("min");
    var max = document.getElementById("max");
    var items = textbox1.value.split(",");
    var obj = min_max(numbers(items));
    min.innerHTML = obj['min'];
    max.innerHTML = obj['max'];
}


function run() {
    var button1 = document.getElementById("button1");
    // TODO: fixme
}

document.addEventListener( "DOMContentLoaded", run);
```

Task 2 (4 Marks)

Using the callstack-microtask-macrotask table

1. illustrate the execution of the following JavaScript program,
2. explain what will be printed on the console output
3. if the program leads to a non-termination, just show one cycle of execution.

```
1:  import EventEmitter from 'events';
2:  const ev1 = new EventEmitter();
3:  const ev2 = new EventEmitter();
4:  let count = 0;
5:  let promise1 = new Promise( (resolve, reject) => {
6:      resolve(count);
7:  })
8:  let promise2 = new Promise( (resolve, reject) => {
9:      resolve(count);
10: })
11: function foo(x) {
12:     return new Promise((resolve, reject) => {
13:         if (x > 10) {
14:             resolve();
15:         } else if (x % 2 == 0) {
16:             ev1.emit('run', ++x);
17:         } else {
18:             ev2.emit('run', ++x);
19:         }
20:     })
21: }
22: ev1.on('run', (data) => {
23:     console.log(`data ${data} received by ev1`);
24:     promise2.then(foo(data));
25: })
26: ev2.on('run', (data) => {
27:     console.log(`data ${data} received by ev2`);
28:     promise1.then(foo(data));
29: })
30:ev2.emit('run', count);
```

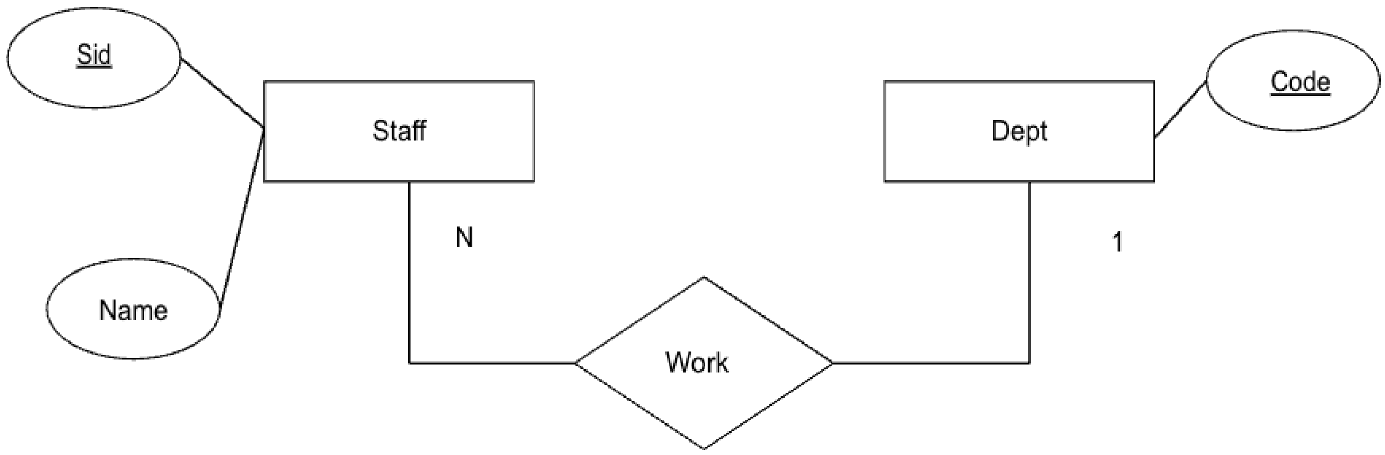
The first few steps of the execution is given as follows to help you get started.

program counter (line num)	call stack	micro queue	promises	macro queue	event reg	console output
5	[main()]	[]	{promise@5}	[]	{}	
8	[main()]	[]	{promise@5, promise@8}	[]	{}	
22	[main()]	[]	{promise@5, promise@8}	[]	{ ev1.run:function@22 }	
26	[main()]	[]	{promise@5, promise@8}	[]	{ ev1.run:function@22,	

program counter (line num)	call stack	micro queue	promises	macro queue	event reg	console output
					ev2.run:function@26 }	
30	[main()]	[]	{promise@5, promise@8}	[function@26(0)]	{ ev1.run:function@22, ev2.run:function@26 }	
eof	[]	[]	{promise@5, promise@8}	[function@26(0)]	{ ev1.run:function@22, ev2.run:function@26 }	

Task 3 (4 marks)

Using MongoDB document database, give a logical design of the ER diagram with the Staff and Dept entities and Work relationship



implement a simple API web-app with the following end-points

1. add dept

   ```
   http://localhost:3000/dept/add/hr
   ```

   yields

```
{"code":"hr","_id":"6478a5a866394647f94f4021"}
```

2. add staff

   ```
   http://localhost:3000/staff/add/1/aaron/hr
   ```

   yields

   ```
   {"id":"1","name":"aaron","dept":"hr","_id":"6478a6de67e208e3a7764c43"}
   ```

3. find all deptartments

   ```
   http://localhost:3000/dept/all/
   ```

   yields

   ```
   [{"code":"hr"}]
   ```

4. find all staffs

   ```
   http://localhost:3000/staff/all/
   ```

   yields

   ```
   [{"id":"1","name":"aaron","dept":"hr"}]
   ```

5. find all depts with staffs

   ```
   http://localhost:3000/dept/all/withstaff/
   ```

   yields

   ```
   [{"code":"hr","staffs":[{"id":"1","name":"aaron","dept":"hr"}]}]
   ```