WEB322 Test 2

Name: _____

Instructions: Complete all questions in the spaces provided. This quiz is worth 10% of your final mark and you will have exactly 30 minutes to complete it.

Question 1 (10 Marks):

For each multiple-choice question, identify the correct answer for the given question. There is only **one** correct answer provided for each question:

1) When using function constructor to create a JavaScript object, which keyword or modifier should be used to initialize property values:

- ☐ const
- ☐ let
- ☐ this
- ☐ var

2) To create a new object with the specified prototype object and properties, use method:

- ☐ Object.clone()
- ☐ Object.create()
- ☐ Object.inheritance()
- ☐ Object.assign()

3) The following method is used to incorporate middleware in the application:

- ☐ app.middleware()
- ☐ app.incorporate()
- ☐ app.responseCycle()
- ☐ app.use()

4) To ensure that the contents of the folder "public" are treated as "static" files, we use the middleware:

- ☐ express.fileTree("public")
- ☐ express.staticList("public")
- ☐ express.files("public")
- ☐ express.static("public")

5) The following method is used to send the client to a new path / url:

- ☐ res.redirect()
- ☐ res.newPath()
- ☐ res.url()
- ☐ res.path()

6) The following method is used to view a specific header on the request (using the req object):

- ☐ req.getHeader()
- ☐ req.get()
- ☐ req.headers()
- ☐ req.meta()

7) Before we can begin a debug session in Visual Studio code, we must generate a:

- ☐ config.json file
- ☐ launch.json file
- ☐ debug.json file
- ☐ run.json file

8) The following method is used to send a specific HTTP status code in the response:

- ☐ res.setStatus()
- ☐ res.code()
- ☐ res.status()
- ☐ res.statusCode()

9) Error handling middleware callback functions are defined with the following parameters:

- ☐ (err, req, res, next)
- ☐ (proc, err, req, res)
- ☐ (use, err, req, res)
- ☐ (next, req, res, type)

10) The following method processes a HTTP GET request to a matching path, using a specified callback function:

- ☐ app.request()
- ☐ app.getRequest()
- ☐ app.get()
- ☐ app.request.get()

Question 2 (4 Marks):

Write JavaScript code to convert user input to a positive number and assign the result to variable **num** by calling the **toPositiveNumber** function. If the **input** is invalid, console log error message, e.g. " Error: the input is not a number!".

```
function toPositiveNumber(x) {
   if(isNaN(x)) { throw new Error("not a number!") };
   if(Number(x) <= 0) { throw new Error("0 or a negative number!") };
   return Number(x);
}

let num = NaN, input = document.getElementById("#input").value;
try{
   num = toPositiveNumber(input);
}catch(ex){
   console.log("Error: the input is " + ex.message);
}
```

Question 3 (7 Marks):

Write a JavaScript function named **guessMyTestMark**. In the function, a random mark between 0 and 100 will be generated by using `Math.floor(Math.random() * 100)`. The function returns a new "Promise" that will "reject" with message "is too bad" if the mark if below 50 and "resolve" with the mark otherwise. Then call the function to console log the "resolved" mark or the "rejected" message.

```
function guessMyTestMark() {
   var mark = Math.floor(Math.random() * 100);
   return new Promise((resolve, reject) =>{
      if (mark < 50) reject ("is too bad");
      else resolve(mark);
   });

}

guessMyTestMark()
   .then( mark => console.log("My mark is ", mark)) // e.g. My mark is 85
   .catch( ex => console.log("The result", ex));    // e.g. The result is too bad
```

Question 4 (11 Marks):

Given a server.js file that uses express (i.e. var app = express();), write code to respond to the routes below as specified in the description. **NOTE**: all callback functions **must be written** in the new ES6 **"Arrow Function"** syntax:

a) Assuming our server.js is using the "**path**" module, match the "**GET**" route "**/students/welcome**" and send the file "**welcome.html**" (located at the root of your application folder). (HINT: do not forget to include **__dirname** in your solution) [3 Marks]

```
app.get("/students/welcome",(req,res)=>{
    res.sendFile(path.join(__dirname + "/welcome.html"));
});
```

b) Match the "**GET**" route "**/studentList**" and return a JSON formatted string: **{message: "all students"}.** Additionally, this route will support optional queries: [3 marks]

- "/**studentList**?**program**=**CPA**" which returns a JSON formatted string: **{message: CPA}**
- "/**studentList**?**school**=**ICT**" which returns a JSON formatted string: **{message: ICT }**

```
app.get("/studentList", (req, res) => {
  if (req.query.program) {
      res.json({message: req.query.program });
  } else if (req.query.school) {
      res.json({message: req.query.school});
  } else {
      res.json({message: "all students"});
  }
});
```

c) Match the "**GET**" route "**/student/studentID**" and return a JSON formatted string: **{message: studentID}** where **studentID** is the value, **e.g. 041066050,** passed in the **url.** [2 Marks]

```
app.get("/student/:studNum", (req, res) => {
      res.json({message: req.params.studNum});
});
```

d) Match the "route" that is invoked if **no other paths are matched** (you may assume that it is located below the other route definitions). Ensure that a **404 status code** and the **plain text** message: "**Page Not Found**" are send back. [3 Marks]

```
app.use((req, res) => {
  res.status(404).send("Page Not Found");
});
```