Quiz: Testing

Question 1

1/1 point (graded)

Which properties are present in Integration testing, but are NOT present in System testing?
Select all that apply.

- [x] Runs quickly and frequently
- [ ] Tests the entire system, from top-to-bottom
- [ ] Uses real data from client
- [x] Low-level form of testing
- [ ] High-level form of testing
- [x] Tests the interface between several modules

✔

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 2

1/1 point (graded)

Which of these properties is the major advantage of unit tests over integration tests?

- Fast
- Reliable
- Isolate failures ✔
- Simulate users

Submit You have used 1 of 2 attempts

✔ Correct (1/1 point)

Question 3

1/1 point (graded)

Consider the function that opens a file on your computer.

```
open(path : String, callback : function) throws IOException
```

```
let expected = () => { open(undefined, (err, res) => { if(err) throw err; return res }) }
```

How would you best write a one-line assertion to test the result of `expected`?

- `expect(open(undefined, ()=>{if(err) throw err; return res})).to.throw(Exception);`
- `expect(expected).to.not.throw();`
- `expect(expected).to.throw(IOException);` ✔
- `expect(expected).to.be.undefined;`

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

For Question 4, consider the function below from vscode's label.ts:

```
export function tildify(path: string, userHome: string): string {
   if (path && (platform.isMacintosh || platform.isLinux) &&
isEqualOrParent(path, userHome, !platform.isLinux)) {
    path = `~${path.substr(userHome.length)}`;
  }

   return path;
}
```

Question 4

1/1 point (graded)

Imagine that you write two test cases for the function `tildify`, as follows:

- `path = "/"`, `platform.isMacintosh` is true, and `isEqualOrParent(...)` evaluates to true
- `path = null`

Indicate on the 4-way rectangle venn diagram below which types of coverage you have achieved. (i.e. if only branch and line coverage are fulfilled, then place the green dot in the rectangle of their intersection.)

✔

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

For Questions 5 and 6, consider the following function, which may be used as a helper to manage item or person quantity in an area. It returns a function based on whether or not the currentCapacity exceeds maxCapacity.

```
public function getCapacityRegulator(currentCount: number) : (n : number) =>
number {
    let applyThis;
    let maxCapacity = this.maxCapacity;

    if (this.specialEvent) {
        maxCapacity = this.eventCapacity;
    }

    if (currentCount < maxCapacity) {
        applyThis = (dec : number) => { return dec--;}
    } else {
        applyThis = (inert : number) => { return inert; }
    }

    return applyThis;
}
```

Question 5

1/1 point (graded)

Select the minimum combination of test cases that would definitely achieve full **branch** coverage for `getCapacityRegulator`.

- ☑ `specialEvent` == true and currentCount > eventCapacity

- [ ] `specialEvent` == true and currentCount < eventCapacity
- [x] `specialEvent` == false and currentCount < this.maxCapacity
- [ ] `specialEvent` == false and currentCount > eventCapacity

✔

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 6

1/1 point (graded)

In addition to the test cases from the previous question, which tests would you have to add to achieve full **path** coverage for `getCapacityRegulator`?

- [ ] `specialEvent` == true and currentCount > eventCapacity
- [x] `specialEvent` == true and currentCount < eventCapacity
- [ ] `specialEvent` == false and currentCount < this.maxCapacity
- [x] `specialEvent` == false and currentCount > this.maxCapacity

✔

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

Question 7

1/1 point (graded)

What is something you can do to improve isolateability in your code?

- [ ] Write a programmatic trigger for things like MouseEvents or KeyEvents
- [x] Write mocks of external services that can predictably give you a certain result ✔
- [ ] Refactor your methods to take in external services as parameters
- [ ] Add a shell script that executes all your test suites

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)

For Question 8, consider the following test for a method which extracts the extension from a path.

```
test('extname', () => {
  assert.equal(paths.extname('far.boo'), '.boo');
  assert.equal(paths.extname('far.b'), '.b');
  assert.equal(paths.extname('far.'), '.');
  assert.equal(paths.extname('far.boo/boo.far'), '.far');
  assert.equal(paths.extname('far.boo/boo'), '');
});
```

Question 8

1/1 point (graded)
Which of the following extra cases should you include to improve coverage of the valid input space?

- [x] 'foo.foo.bar'
- [x] null

- [x] undefined
- [x] '!.bar'
- [x] ''

Submit You have used 1 of 2 attempts

✔ Correct (1/1 point)

For Question 9, consider the following snippet from vscode:

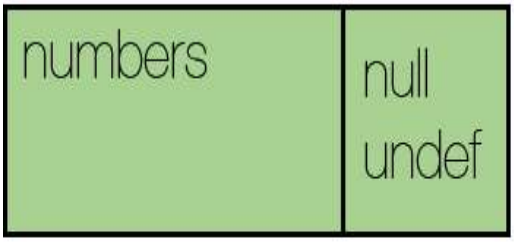
export class RGBA {

```
// Red: integer in [0-255]
readonly r: number;

// Green: integer in [0-255]
readonly g: number;

// Blue: integer in [0-255]
readonly b: number;

// Alpha: integer in [0-255]
readonly a: number;
```

```
constructor(r: number, g: number, b: number, a: number = 255) {
      this.r = Math.min(255, Math.max(0, r)) | 0;
      this.g = Math.min(255, Math.max(0, g)) | 0;
      this.b = Math.min(255, Math.max(0, b)) | 0;
      this.a = Math.min(255, Math.max(0, a)) | 0;
}
```

Then consider these four images depicting possible input and output spaces:

A



B

C

D

Questions 9

1/1 point (graded)

Which of the images above correctly describes the **input** space for the constructor? (Input spaces are considered to be the green area within the image.)

A ▾ ✔

Submit You have used 1 of 1 attempt

✔ Correct (1/1 point)