Jasmin & Karma (Assignment-1)

1. Brief Description about unit testing and functional testing and it's benefit in project, from developer perspective.

Unit testing:

A type of software testing in which individual units/components of software are tested to determine their usefulness. Unit testing is carried out by developer during development phase of a software.

Advantages of unit testing:

- Bugs can be detected at early stage, and development team can take care without any major complications.
- Improves code design.
- Developers can look at tests and can conclude how it works (functionality of code). So, it also acts as documentation.

Functional testing:

A type of software testing in which the software is tested against the specifications and functional requirements. It ensures the specifications and functional requirements are satisfied. It is not concerned about the source code. Each functionality is tested by providing appropriate test input and comparing expected output with actual output.

Advantages of functional testing :

- Ensure bug free product delivery
- Ensures high quality product delivery
- No assumption about the structure of the system
- This testing is focused on the specifications as the customer usage.

2 . Where and why, you need unit testing in your project with 10 examples.

It is used to verify that all functions and sections are working as required. To find and fix bugs.

1. . // Arrange

   let count = 10;

   // Act

   const newCount = increment(count);

```
// Assert
 expect(newCount).toBe(11);
```

2.
```
describe('FileUploadComponent', () => {
 let component: FileUploadComponent;
 let fixture: ComponentFixture;
  beforeEach(() => {
        TestBed.configureTestingModule({
              declarations: [ FileUploadComponent ]
              });
        fixture = TestBed.createComponent(FileUploadComponent);
        component = fixture.componentInstance;
         fixture.detectChanges();
 });
  it('should create file uploader component', () => {
        expect(component).toBeTruthy();
        });
 });
```

3.
```
describe("ParagraphComponent", () => {
 let component: ParagraphComponent;
 let fixture: ComponentFixture;
         let dbgElement: DebugElement;
        let element: HTMLElement;
```

```
    beforeEach(() => {
    TestBed.configureTestingModule({
                declarations: [ParagraphComponent]
        });
        fixture = TestBed.createComponent(ParagraphComponent);
        component = fixture.componentInstance;

        dbgElement = fixture.debugElement.query(By.css(".phone"));
        element = dbgElement.nativeElement;

        fixture.detectChanges();
    });

    it("paragraph should contain default message", () => {
        expect(element.innerText).toContain("not specified");
    });
    it("paragraph should contain phone number", () => {
        component.phone = "0021000111";
        fixture.detectChanges();
        expect(element.innerText).toContain("0021000111");
    });
});
```

4.
```
describe('MinutesToHoursPipe', () => {
        const pipe = new MinutesToHoursPipe();
        it('return same value if undefined', () => {
                expect(pipe.transform(undefined)).toBe(undefined);
        });
    it('transform minutes to hours and minutes string', () => {
                 const testData = {
                        1: '1 minute',
                        30: '30 minutes',
                         60: '1 hour',
                        90: '1 hour and 30 minutes',
                         97: '1 hour and 37 minutes',
                        120: '2 hours',
                        121: '2 hours and 1 minute',
                        131: '2 hours and 11 minutes'
                };
                Object.keys(testData).forEach((key) => {
                expect(pipe.transform(key)).toBe(testData[key]);
                });
        });
});
```

5.
```
describe('Company.Model', () => {
        it('should create an instance', () => {
                expect(new Company.Model()).toBeTruthy();
        });
});
```

6.
```
describe('Company.Model', () => {
        it('should create an instance', () => {
                expect(new Company.Model()).toBeTruthy();
        });
});
```

7.
```
it ('should check incremented value is greater than zero', ()=> {
        let counterComponent: CounterComponent = new CounterComponent();
        const curCounterValue = counterComponent.increaseCounter();
        expect(curCounterValue).toBeGreaterThan(0);
});
```

8.
```
it ('should increment if input is positive', ()=> {
        const counterResult = counterParameter(1);
        expect(counterResult).toBe(2);
}) ;
```

9.
```
import { Component, OnInit } from '@angular/core';
        @Component({
        selector: 'app-hotel,
        templateUrl: './hotel.component.html',
        styleUrls: ['./hotel.component.css']
})
```

```
export class HotelComponentimplements OnInit {
        title = "World's best hotels!"
}
```

Test-Case:

```
 it(`should have a title ' World's best hotels!'`, async(() => {
        fixture = TestBed.createComponent(HOtelComponent);
        component = fixture.debugElement.componentInstance;
        expect(component.title).toEqual(World's best hotels!');
}));
```

10.
```
function throwsError() {
     throw new TypeError("A type error");
}
```

Test-Case:

```
it('it should throw an exception', function () {
     expect(throwsError).toThrow();
});
```