1.Write a brief description of unit testing and functional testing and its benefits from a developer perspective.

Unit Testing is defined as testing small modules or components of a large application. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development(coding phase) of an application by the developers.

Advantages:

1.Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.

2.Reduces the Cost of Testing as defects are captured in the very early phase.

3.Improves design and allows better refactoring of code.

4.Unit Tests, when integrated with the build give the quality of the build as well.

Functional Testing is defined as software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Advantages:

1.It produces a defect-free product.

2.It ensures that the customer is satisfied.

3.It ensures that all requirements are met.

4.It ensures that the software/ product works as expected.

2. Where and why you need unit testing in your project give me 10 examples. Where and why: Unit tests are also especially useful when it comes to refactoring or writing a piece of code. If you have good unit test coverage, you can refactor with confidence. Without unit tests, it is often hard to ensure that you didn't break anything. Code Snap:

1:

```
it(`should have as title 'AngularApp'`, () => {
const fixture = TestBed.createComponent(AppComponent);
const app = fixture.componentInstance;
 expect(app.title).toEqual('AngularApp');
});
```

2:
```
it('should create the app', () => {
const fixture = TestBed.createComponent(AppComponent);
 const app = fixture.componentInstance;
 expect(app).toBeTruthy();
});
```

3:
```
it('should create', () => {
expect(component).toBeTruthy();
});
```

4:
```
describe('Company.Model', () => {
it('should create an instance', () => {
expect(new Company.Model()).toBeTruthy();
});
});
```

5:
```
it(`should have as title 'UnitTestExample'`, () =>{
const fixture = TestBed.createComponent(AppComponent);
const app = fixture.componentInstance;
expect(app.title).toEqual('UnitTestExample');
});
```

6:
```
it('should change title to Unit Test App', async(()=> {
const fixture = TestBed.createComponent(AppComponent);
fixture.nativeElement.querySelector('button').click();
fixture.detectChanges();
expect(fixture.nativeElement.querySelector('h1');
textContent).toEqual('Unit Test App');
}));
```

7:
```
it('should have a defined component', () => {
expect(component).toBeDefined();
});
```

8:
```
it ('should check incremented value is greater than zero', ()=> {
let counterComponent: CounterComponent = new CounterComponent();
const curCounterValue = counterComponent.increaseCounter();
expect(curCounterValue).toBeGreaterThan(0);
});
```

9:
```
it ('should check decremented value is less than zero', ()=> {
let counterComponent: CounterComponent = new CounterComponent();
const curCounterValue = counterComponent.decreaseCounter();
expect(curCounterValue).toBeLessThan(0);
});
```

10:
```
it ('should increment if input is positive', ()=>{
const counterResult = counterParameter(1);
expect(counterResult).toBe(2);
});
```