Q1: Which testing tool is good for your project and why (write full description and give some example)?

Ans:

Jasmine is the recommended testing framework. Angular CLI comes by default with Jasmine and Karma as the test runner. It generates test files at component creation, collects unit tests, runs karma, and displays results on a web page.

Jasmine is a Behavior-Driven Development(BDD) framework that provides a clean, obvious Syntax for writing test cases by making the process easy. It has a great community as well as impressive documentation that enables a smooth learning curve.

Karma is the default test runner for Angular. It includes the option to test the code on several browsers and even devices to ensure the smooth operation of the code on the supported platforms.

Among its advantages, the main ones are that it doesn't rely on other JavaScript frameworks, and it also doesn't require a DOM. One of the main goals of Jasmine is to provide an easy syntax, allowing you to write tests more easily. Jasmine is the default test framework used with Angular. It ships with Angular CLI by default. With such low friction required to use it, it's not surprising so many people adopt it.

Ex:

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();
});
```

Q2: Write five sample unit test cases in Karma or Jest.

1:

```
describe('Company.Model', () => {

  it('should create an instance', () => {

    expect(new Company.Model()).toBeTruthy();

  });

});
```

2:

```
it(`should have as title ‘UnitTestExample’`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual(‘UnitTestExample’);

});
```

3:

```
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');
}));
```

4:

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

5:

```
it ('should check incremented value is greater than zero', ()=> {
 let counterComponent: CounterComponent = new CounterComponent();
```

```
const curCounterValue = counterComponent.increaseCounter();

expect(curCounterValue).toBeGreaterThan(0);

});
```