Marvellous Infosystems: Angular Assignment 15

Name: Shrirang Jagdish Nikam

Enrollment No: 396AM_Shrirang

1. Create angular application which contains multiple methods as

CountCapital() - Which returns number of capital letters from string.

CheckPassword() - Which checks

ArrayAddition() - Which accept array of integers and returns addition of all elements.

Write test cases to test above two methods in spec.ts file.

Answer:

Math.service.ts

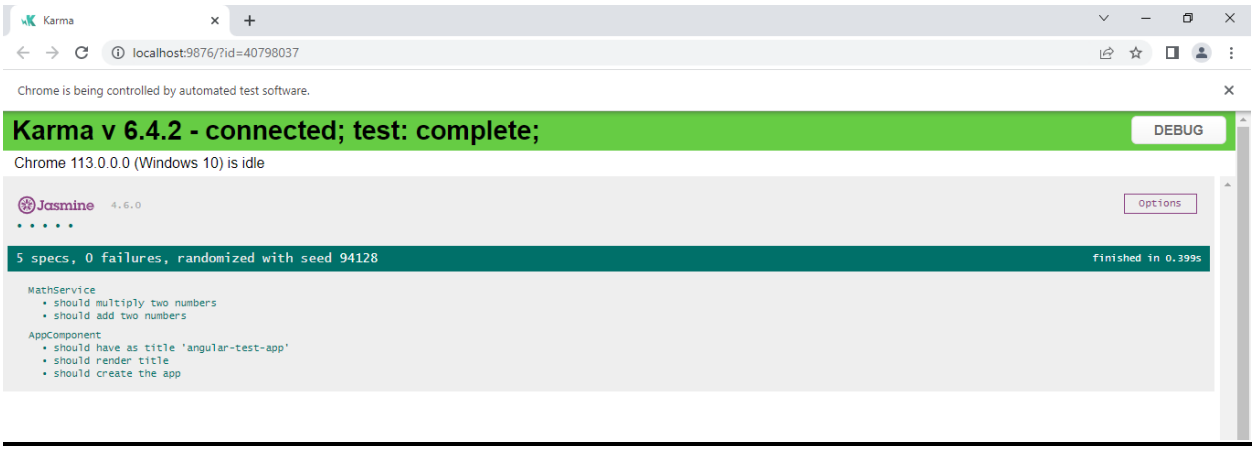
```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class MathService {
  addNumbers(a: number, b: number): number {
    return a + b;
  }

  multiplyNumbers(a: number, b: number): number {
    return a * b;
  }
}
```

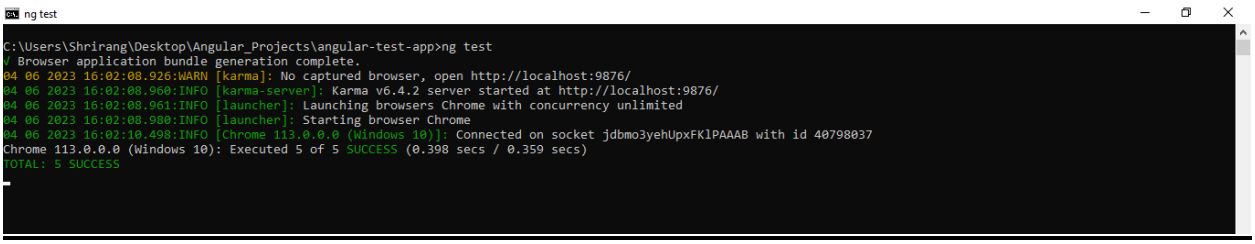
Math.service.spec.ts:

```
import { TestBed } from '@angular/core/testing';

import { MathService } from './math.service';

describe('MathService', () => {
  let service: MathService;

  beforeEach(() => {
```

```
    TestBed.configureTestingModule({});
    service = TestBed.inject(MathService);
  });

  it('should add two numbers', () => {
    const result = service.addNumbers(2, 3);
    expect(result).toBe(5);
  });

  it('should multiply two numbers', () => {
    const result = service.multiplyNumbers(2, 3);
    expect(result).toBe(6);
  });
});
```

Output:





2.Create angular application which creates two custom pipe named as MyAdd & MyMult.

MyAdd pipe is used to add two numbers and MyMult is used to multiply two numbers. (From previous assignment)

Write test cases to test working of both the pipes in its corresponding spec.ts file.

Answer:

My-add.pipe.ts:

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'myAdd'
})
export class MyAddPipe implements PipeTransform {
  transform(a: number, b: number): number {
    return a + b;
  }
}
```

My-add.pipe.spec.ts:

```
import { MyAddPipe } from './my-add.pipe';

describe('MyAddPipe', () => {
  let pipe: MyAddPipe;

  beforeEach(() => {
    pipe = new MyAddPipe();
  });

  it('should add two numbers', () => {
    const result = pipe.transform(2, 3);
    expect(result).toBe(5);
  });
});
```

My-mult.pipe.ts:

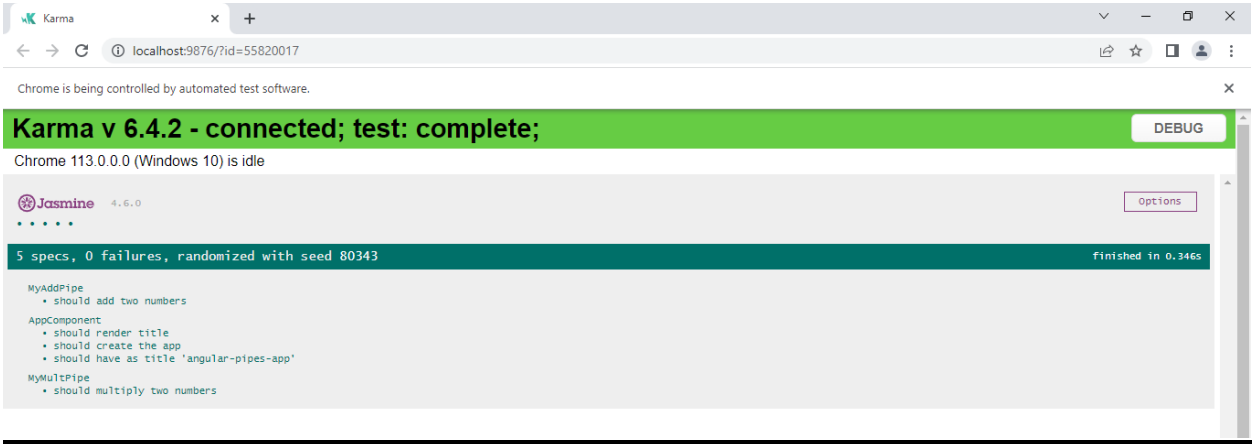
```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
```

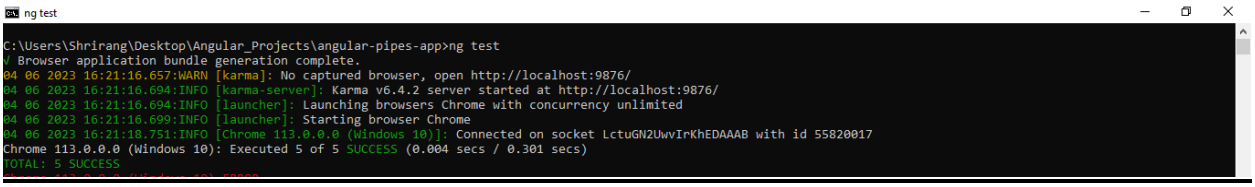
```
  name: 'myMult'
})
export class MyMultPipe implements PipeTransform {
  transform(a: number, b: number): number {
    return a * b;
  }
}
```

My-mult.pipe.spec.ts:

```
import { MyMultPipe } from './my-mult.pipe';

describe('MyMultPipe', () => {
  let pipe: MyMultPipe;

  beforeEach(() => {
    pipe = new MyMultPipe();
  });

  it('should multiply two numbers', () => {
    const result = pipe.transform(2, 3);
    expect(result).toBe(6);
  });
});
```

Output:





3.Create angular application which creates one custom pipe named as MarvellousChk. This

custom pipe accept one integer as a value and parameter can be Prime,Perfect,Even,Odd.

Depends on the parameter we have to check status of the number and return the result

accordingly. (From previous assignment)

Write test cases to test working of both the pipes in its corresponding spec.ts file.

Answer:

Marvelous-chk.pipe.ts:

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'marvellousChk'
})
```

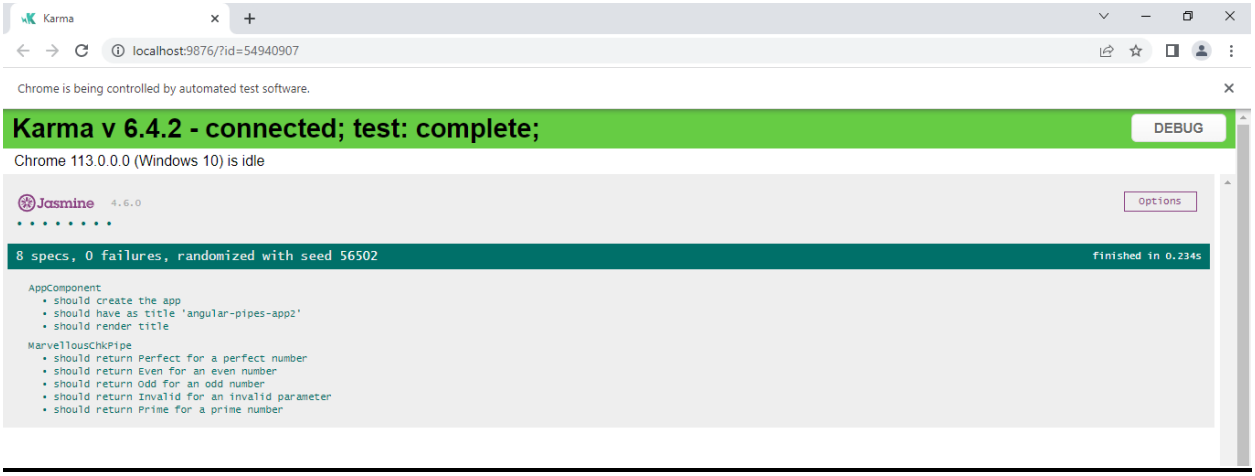
```
export class MarvellousChkPipe implements PipeTransform {
  transform(value: number, parameter: string): string {
    switch (parameter) {
      case 'Prime':
        if (this.isPrime(value)) {
          return 'Prime';
        }
        break;
      case 'Perfect':
        if (this.isPerfect(value)) {
          return 'Perfect';
        }
        break;
      case 'Even':
        if (value % 2 === 0) {
          return 'Even';
        }
        break;
      case 'Odd':
        if (value % 2 !== 0) {
          return 'Odd';
        }
        break;
    }
    return 'Invalid';
  }

  private isPrime(num: number): boolean {
    if (num <= 1) {
      return false;
    }
    for (let i = 2; i < num; i++) {
      if (num % i === 0) {
        return false;
      }
    }
    return true;
  }

  private isPerfect(num: number): boolean {
    let sum = 0;
    for (let i = 1; i < num; i++) {
      if (num % i === 0) {
        sum += i;
      }
```

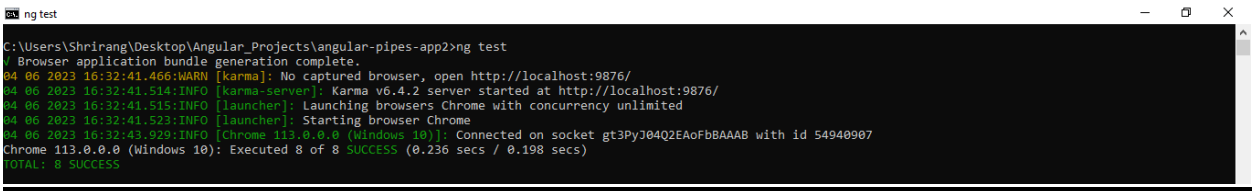
```
    }
    return sum === num;
  }
}
```

Marvelous-chk.pipe.spec.ts:

```
import { MarvellousChkPipe } from './marvellous-chk.pipe';

describe('MarvellousChkPipe', () => {
  let pipe: MarvellousChkPipe;

  beforeEach(() => {
    pipe = new MarvellousChkPipe();
  });

  it('should return Prime for a prime number', () => {
    const result = pipe.transform(5, 'Prime');
    expect(result).toBe('Prime');
  });

  it('should return Perfect for a perfect number', () => {
    const result = pipe.transform(6, 'Perfect');
    expect(result).toBe('Perfect');
  });

  it('should return Even for an even number', () => {
    const result = pipe.transform(4, 'Even');
    expect(result).toBe('Even');
  });

  it('should return Odd for an odd number', () => {
    const result = pipe.transform(3, 'Odd');
    expect(result).toBe('Odd');
  });

  it('should return Invalid for an invalid parameter', () => {
    const result = pipe.transform(5, 'InvalidParam');
    expect(result).toBe('Invalid');
  });
});
```

Output:


Karma
localhost:9876/?id=54940907
Chrome is being controlled by automated test software.
Karma v 6.4.2 - connected; test: complete;
DEBUG
Chrome 113.0.0.0 (Windows 10) is idle
Jasmine 4.6.0
Options
8 specs, 0 failures, randomized with seed 56502
finished in 0.234s
AppComponent
• should create the app
• should have as title 'angular-pipes-app2'
• should render title
MarvellousChkPipe
• should return Perfect for a perfect number
• should return Even for an even number
• should return Odd for an odd number
• should return Invalid for an invalid parameter
• should return Prime for a prime number


ng test
C:\Users\Shrirang\Desktop\Angular_Projects\angular-pipes-app2>ng test
√ Browser application bundle generation complete.
04 06 2023 16:32:41.466:WARN [karma]: No captured browser, open http://localhost:9876/
04 06 2023 16:32:41.514:INFO [karma-server]: Karma v6.4.2 server started at http://localhost:9876/
04 06 2023 16:32:41.515:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
04 06 2023 16:32:41.523:INFO [launcher]: Starting browser Chrome
04 06 2023 16:32:43.929:INFO [Chrome 113.0.0.0 (Windows 10)]: Connected on socket gt3PyJ04Q2EAoFbBAAAB with id 54940907
Chrome 113.0.0.0 (Windows 10): Executed 8 of 8 SUCCESS (0.236 secs / 0.198 secs)
TOTAL: 8 SUCCESS