Scripting Lab Assignment

Name – Shashwat sinha

Section – C

Registration number – 201900358

Semester – 5

Create a calculator app using Angular which is capable of performing following operations:

1. Addition of two numbers
2. Subtraction of two numbers
3. Multiplication of two numbers
4. Division of two numbers
5. Factorial of a number
6. Checking if a given number is Prime or not

CODE:

Basic setup:

below command in the command prompt at the desired location where you want to create a project.

```
ng new angular-calculator-app
```

Below is the command for going to the project directory:

```
cd angular-calculator-app
```

Below is the command to install bootstrap in the project.

```
npm install –save bootstrap@latest
```

Below is the code of the angular.json file:

```
"styles": [
 "src/styles.css",
 "node_modules/bootstrap/dist/css/bootstrap.min.css"
 ],
```

below is the code which we need to include in our app.component.html for creating the main

display and sub-display.

```
 <div class="maindisplay">
 <div class="subdisplay">{{ subDisplayText }}</div>
 {{ mainDisplayText }}
 </div>
```

Below is the complete code of app.component.html.

```
<div class="container">
 <div class="row">
 <div class="col-md-4"></div>
 <div class="col-md-4">
 <div class="base">
 <div class="maindisplay">
```

```
<div class="subdisplay">{{ subDisplayText }}</div>

{{ mainDisplayText }}

</div>

<div class="keypad">

<table style="width: 100%;">

<tr>

<td class="keys ackey" colspan="3" (click)="allClear()">AC</td>

<td class="keys opkey" colspan="1" (click)="pressKey('/')">/</td>

</tr>

<tr>

<td class="keys numkey" (click)="pressKey('7')">7</td>

<td class="keys numkey" (click)="pressKey('8')">8</td>

<td class="keys numkey" (click)="pressKey('9')">9</td>

<td class="keys opkey" (click)="pressKey('x')">x</td>

</tr>

<tr>

<td class="keys numkey" (click)="pressKey('4')">4</td>

<td class="keys numkey" (click)="pressKey('5')">5</td>

<td class="keys numkey" (click)="pressKey('6')">6</td>

<td class="keys opkey" (click)="pressKey('-')">-</td>

</tr>

<tr>

<td class="keys numkey" (click)="pressKey('3')">3</td>

<td class="keys numkey" (click)="pressKey('2')">2</td>
```

```
 <td class="keys numkey" (click)="pressKey('1')">1</td>

 <td class="keys opkey" (click)="pressKey('+')">+</td>

 </tr>

 <tr>

 <td colspan="2" class="keys numkey" (click)="pressKey('0')">0</td>

 <td class="keys numkey" (click)="pressKey('.')">.</td>

 <td class="keys equalkey" (click)="getAnswer()">=</td>

 </tr>

 </table>

 </div>

 </div>

 </div>

 <div class="col-md-4"></div>

 </div>

</div>
```

Below is the complete code of the app.component.css file.

app.component.css

```
.base {

 background: darkslategray;

 margin-top: 5vh;

 border: 3px solid black;

 width: 100%;
```

```
}

.maindisplay {

 background: lightgrey;

 height: 25vh;

 padding: 5% !important;

 font-size: 4rem;

 text-align: right;

 font-family: Courier, monospace;

 overflow: auto;

}

.subdisplay {

 border-bottom: 1px solid black;

 height: 25%;

 font-size: 2rem;

 overflow: auto;

}

.keypad {

 height: calc(200% / 3);

}

.keys {

 margin: 0;

 height: 20%;

 background: whitesmoke;

 color: grey;
 padding: 5%;

 font-size: 2rem;

 text-align: center;

 cursor: pointer;

 opacity: 0.9;

}

.keys:hover {

 opacity: 1;

}

.ackey {

 color: red;

 background: black;

}

.equalkey {

 color: white;

 background-color: orangered;

}

.numkey {

 color: skyblue;

 background-color: grey;

}

.opkey {

 color: white;

 background-color: black;
```

}

Below is the code for declaring some variables such as mainDisplayText, subDisplayText, first

Operand, etc.

```
subDisplayText = '';

mainDisplayText = '';

operand1: number;

operand2: number;

operator = '';
```

below is the code for pressKey function.

```
pressKey(key: string) {

 if (key === '/' || key === 'x' || key === '-' || key === '+') {

 const lastKey = this.mainDisplayText[this.mainDisplayText.length - 1];

 if (lastKey === '/' || lastKey === 'x' || lastKey === '-' || lastKey === '+') {

 this.operatorSet = true;

 }

 if ((this.operatorSet) || (this.mainDisplayText === '')) {

 return;

 }

 this.operand1 = parseFloat(this.mainDisplayText);
 this.operator = key;

 this.operatorSet = true;

 }

 if (this.mainDisplayText.length === 10) {

 return;

 }

 this.mainDisplayText += key;

 }
```

Below is the code of allClear( ) function:

```
allClear() {

 this.mainDisplayText = '';

 this.subDisplayText = '';

 this.operatorSet = false;

 }
```

below is the function for performing and handling the calculation.

```
getAnswer() {

 this.calculationString = this.mainDisplayText;

 this.operand2 = parseFloat(this.mainDisplayText.split(this.operator)[1]);

 if (this.operator === '/') {

 this.subDisplayText = this.mainDisplayText;

 this.mainDisplayText = (this.operand1 / this.operand2).toString();

 this.subDisplayText = this.calculationString;
```

```
if (this.mainDisplayText.length > 9) {

this.mainDisplayText = this.mainDisplayText.substr(0, 9);

}

} else if (this.operator === 'x') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 * this.operand2).toString();

this.subDisplayText = this.calculationString;

if (this.mainDisplayText.length > 9) {

this.mainDisplayText = 'ERROR';

this.subDisplayText = 'Range Exceeded';

}

} else if (this.operator === '-') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 - this.operand2).toString();

this.subDisplayText = this.calculationString;

} else if (this.operator === '+') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 + this.operand2).toString();

this.subDisplayText = this.calculationString;

if (this.mainDisplayText.length > 9) {

this.mainDisplayText = 'ERROR';

this.subDisplayText = 'Range Exceeded';

}

} else {
 this.subDisplayText = 'ERROR: Invalid Operation';

 }

 this.answered = true;

 }
```

Below is the complete code of the app.component.ts file.

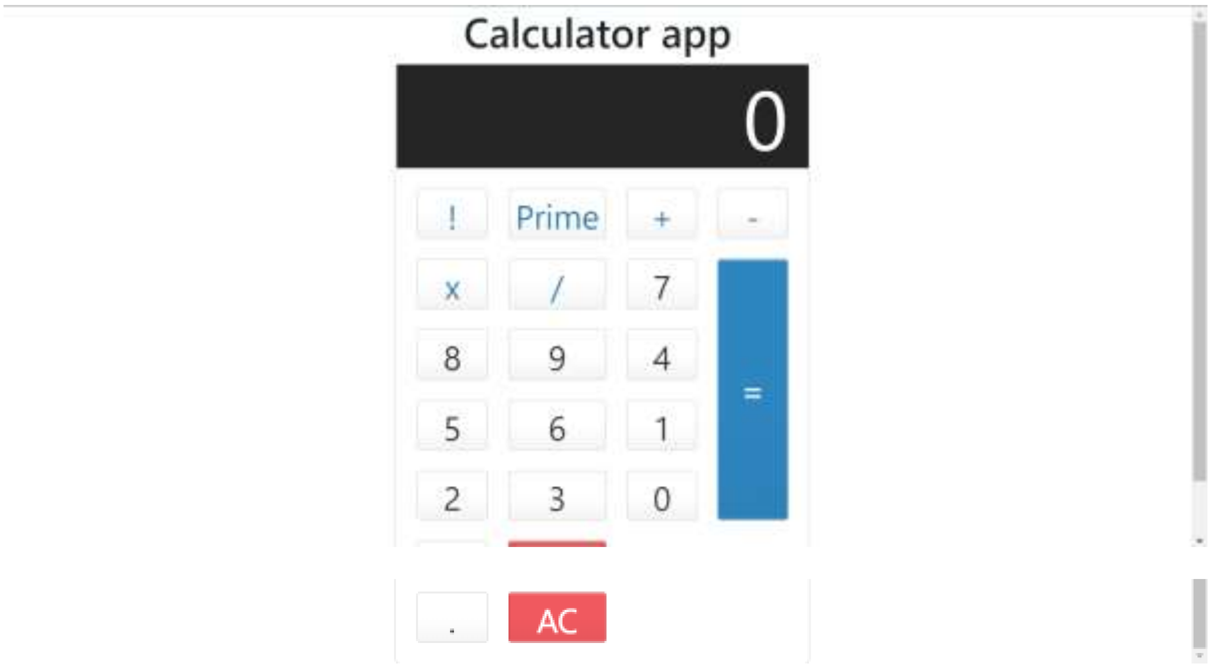
app.component.ts

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

@Component({

 selector: 'app-root',

 templateUrl: './app.component.html',

 styleUrls: ['./app.component.css']

})

export class AppComponent {

 title = 'angular-calculator-app';

 subDisplayText = '';

 mainDisplayText = '';

 operand1: number;

 operand2: number;

 operator = '';

 calculationString = '';

 // This string denotes the operation being performed

 answered = false;
```

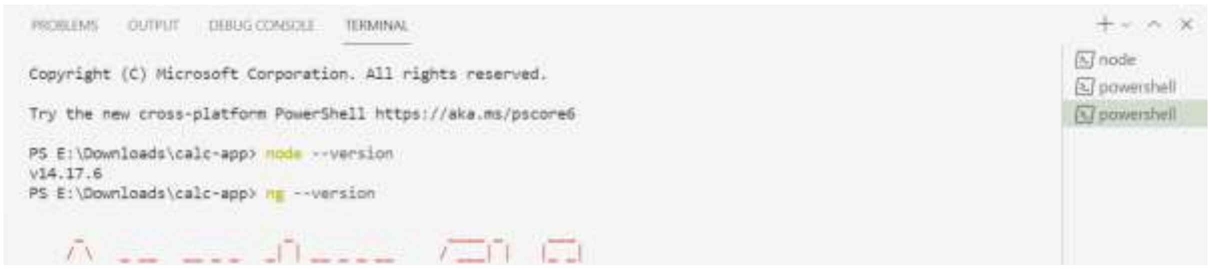
```
// flag to check whether the solution has been processed

operatorSet = false;

pressKey(key: string) {

if (key === '/' || key === 'x' || key === '-' || key === '+') {

const lastKey = this.mainDisplayText[this.mainDisplayText.length - 1];

if (lastKey === '/' || lastKey === 'x' || lastKey === '-' || lastKey === '+') {

this.operatorSet = true;

}

if ((this.operatorSet) || (this.mainDisplayText === '')) {

return;

}

this.operand1 = parseFloat(this.mainDisplayText);

this.operator = key;

this.operatorSet = true;

}

if (this.mainDisplayText.length === 10) {

return;

}

this.mainDisplayText += key;

}

allClear() {

this.mainDisplayText = '';

this.subDisplayText = '';

this.operatorSet = false;
```

```
}

getAnswer() {

this.calculationString = this.mainDisplayText;

this.operand2 = parseFloat(this.mainDisplayText.split(this.operator)[1]);

if (this.operator === '/') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 / this.operand2).toString();

this.subDisplayText = this.calculationString;

if (this.mainDisplayText.length > 9) {

this.mainDisplayText = this.mainDisplayText.substr(0, 9);

}

} else if (this.operator === 'x') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 * this.operand2).toString();

this.subDisplayText = this.calculationString;

if (this.mainDisplayText.length > 9) {

this.mainDisplayText = 'ERROR';

this.subDisplayText = 'Range Exceeded';

}

} else if (this.operator === '-') {

this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 - this.operand2).toString();

this.subDisplayText = this.calculationString;

} else if (this.operator === '+') {
```

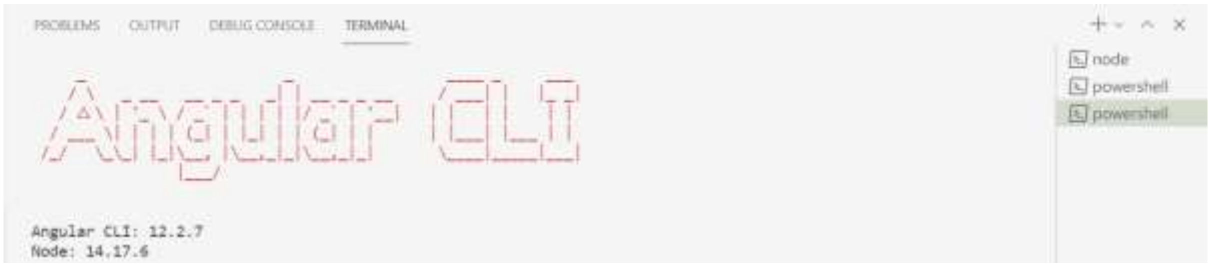
```
this.subDisplayText = this.mainDisplayText;

this.mainDisplayText = (this.operand1 + this.operand2).toString();

this.subDisplayText = this.calculationString;

if (this.mainDisplayText.length > 9) {

this.mainDisplayText = 'ERROR';

this.subDisplayText = 'Range Exceeded';

}

} else {

this.subDisplayText = 'ERROR: Invalid Operation';

}

this.answered = true;

}

}
```



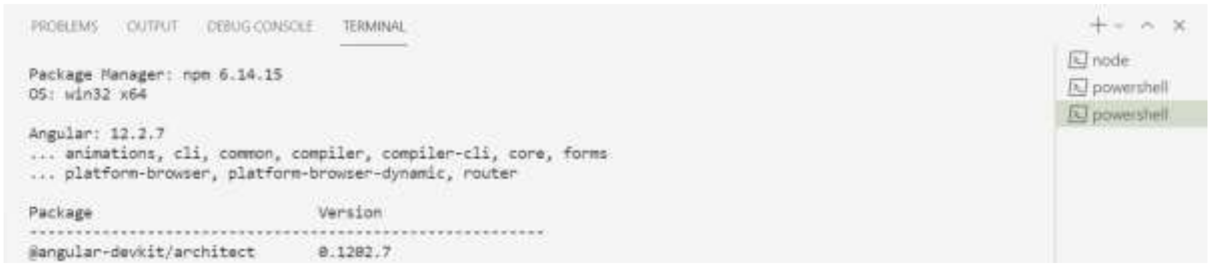
Screenshot of all installations taken in vs code



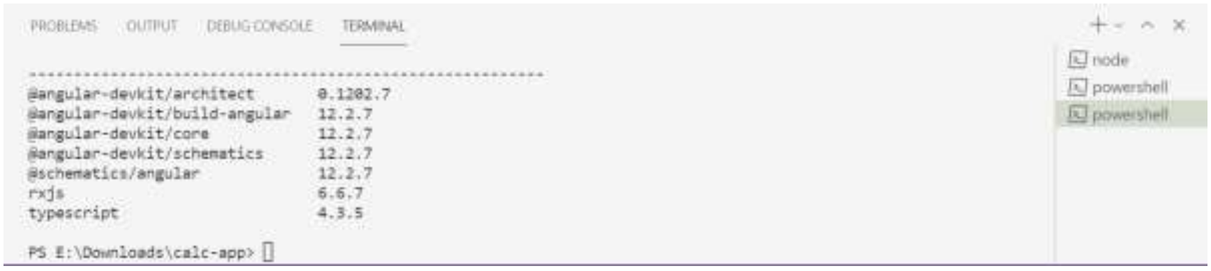
```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\Downloads\calc-app> node --version
v14.17.6
PS E:\Downloads\calc-app> ng --version
```



```
Angular CLI: 12.2.7
Node: 14.17.6
```



```
Package Manager: npm 6.14.15
OS: win32 x64

Angular: 12.2.7
... animations, cli, common, compiler, compiler-cli, core, forms
... platform-browser, platform-browser-dynamic, router

Package                         Version
---------------------------------------------------------
@angular-devkit/architect       0.1202.7
```



```
---------------------------------------------------------
@angular-devkit/architect       0.1202.7
@angular-devkit/build-angular   12.2.7
@angular-devkit/core            12.2.7
@angular-devkit/schematics      12.2.7
@schematics/angular             12.2.7
rxjs                            6.6.7
typescript                      4.3.5

PS E:\Downloads\calc-app> 
```