Slip 1(a) Write an Angular 13 application addition of two numbers using ng-init, ng-model & ng-bind. And also demonstrate ng-show, ng-disabled, ng-click directives on button component.

a) ng new slip1

b) cd slip1

c) open app.component.ts

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


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  num1: number = 0;
  num2: number = 0;
  result: number = 0;
  disableButton: boolean = false;
  showResult: boolean = false;

  add(): void {
    this.result = this.num1 + this.num2;
    this.disableButton = true;  // Disable the button after calculation
    this.showResult = true;      // Show the computed result
  }
}
```

2)app.component.html

```
<div>
    <h2>Add Two Numbers</h2>
 <label>
      First Number:
```

```
      <input type="number" [(ngModel)]="num1">
    </label> <br>
  <label>
      Second Number:
      <input type="number" [(ngModel)]="num2">
    </label>
    <br><br>

    <button (click)="add()" [disabled]="disableButton">
      Compute Sum
    </button>
    <br><br>

    <h3 *ngIf="showResult">
      Result: {{ result }}
    </h3>
</div>
```

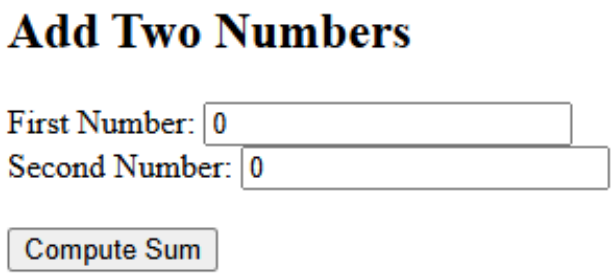
3) app.module.ts

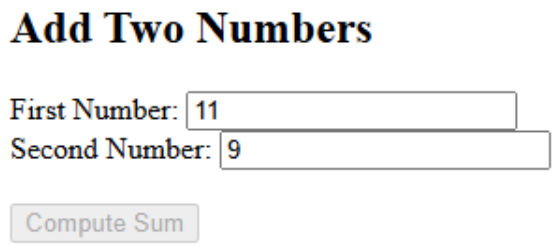
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,FormsModule
  ],
  providers: [],
```

```
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Run project -

ng serve -o





2) Write an AngularJS script to print details of bank (bank name, MICR code, IFC code, address etc.) in tabular form using ng-repeat.

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

```
interface Bank {
name: string;
micr: string;
ifsc: string;
address: string;
}
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
banks: Bank[] = [
{
name: 'HDFC',
micr: '123456789',
ifsc: 'FNB0001234',
address: 'Chinchwad'
},
{
name: 'SBI',
micr: '987654321',
ifsc: 'PTB0009876',
address: 'Pimpri'
},
{
name: 'TJSB',
micr: '987654323',
ifsc: 'TJB0009876',
address: 'Pune'
},
];
```

```
}
<div>
<h2>Bank Details</h2>
<table>
<thead>
<tr>
<th>Bank Name</th>
<th>MICR Code</th>
<th>IFSC Code</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr *ngFor="let bank of banks">
<td>{{ bank.name }}</td>
<td>{{ bank.micr }}</td>
<td>{{ bank.ifsc }}</td>
<td>{{ bank.address }}</td>
</tr>
</tbody>
</table>
</div>
```

Add formsModule in app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
```

```
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**********output-**********

ng serve -o

Bank Details

Bank Name	MICR Code	IFSC Code	Address
HDFC	123456789	FNB0001234	Chinchwad
SBI	987654321	PTB0009876	Pimpri
TJSB	987654323	TJB0009876	Pune

3) Find a company with a workforce greater than 30 in the array (use find by id method)

```
import { Component } from '@angular/core';
interface Company {
id: number;
name: string;
```

```
workforce: number;
}
@Component({
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
companies: Company[] = [ { id: 1, name: 'Alpha Corp', workforce: 25 },
{ id: 2, name: 'Beta Ltd', workforce: 45 },
{ id: 3, name: 'Gamma LLC', workforce: 30 }
];
foundCompany?: Company;
findCompany(): void {
this.foundCompany = this.companies.find(c => c.workforce > 30);
}
}
```

app.component.html

```
<div class="container">
<h2>Company Search by Workforce</h2>
<button (click)="findCompany()">
Find Company with Workforce > 30
</button>
<div *ngIf="foundCompany; else notFound">
<h3>Found Company:</h3>
<p><strong>ID:</strong> {{ foundCompany.id }}</p>
<p><strong>Name:</strong> {{ foundCompany.name }}</p>
<p><strong>Workforce:</strong> {{ foundCompany.workforce }}</p>
</div>
<ng-template #notFound>
<p>No company found with workforce greater than 30.</p>
</ng-template>
</div>
```

**********output-**********

4) Write an Angular to display list of games stored in an array on click of button using ng-click and also demonstrate ng-init, ng-bind directive of AngularJS.

app.component.ts

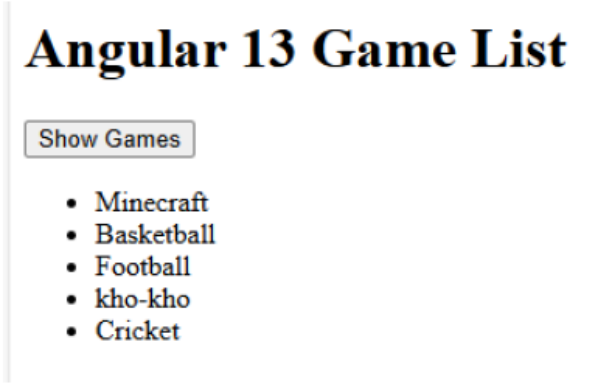
```
import { Component, OnInit } from '@angular/core';
@Component({
selector: 'app-root',
```

```
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']})
export class AppComponent implements OnInit {
title = 'Angular 13 Game List';
games: string[] = [];
showGamesList = false;
constructor() {}
ngOnInit(): void {
// Initialize any default values here
this.games = []; // Empty by default
}
showGames(): void {
this.games = [
'Minecraft',
'Basketball',
'Football',
'kho-kho',
'Cricket'
];
this.showGamesList = true;
}
}
```

app.component.html

```
<h1>{{ title }}</h1>
<button (click)="showGames()">Show Games</button>
<ul *ngIf="showGamesList">
<li *ngFor="let game of games">{{ game }}</li>
</ul>
```

**********output-**********



5) Create a simple Angular component that takes input data and displays it.

ng g c user

user.component.ts

```
import { Component, OnInit, Input } from '@angular/core';
@Component({
selector: 'app-user',
```

```
templateUrl: './user.component.html',
styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
@Input() userName!: string;
@Input() userEmail!: string;
constructor() { }
ngOnInit(): void {
console.log("Username:", this.userName);
console.log("User Email:", this.userEmail);
}
}
```

user.component.html

```
import { Component, OnInit, Input } from '@angular/core';
@Component({
selector: 'app-user',
templateUrl: './user.component.html',
styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {
@Input() userName!: string;
@Input() userEmail!: string;
constructor() { }
ngOnInit(): void {
console.log("Username:", this.userName);
console.log("User Email:", this.userEmail);
}
}
```

app.component.ts

```
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
```

```
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
title = 'inputcomponent';
name = 'Soham kale';
email = 'soham.kale@example.com';
}
```

app.component.html

```
<h2>{{ title }}</h2>
<!-- Passing data to child component -->
<app-user [userName]="name" [userEmail]="email"></app-user>
```

**********output-**********

6) Create Angular 13application that print the name of students who got 85% using filter and map method.

app.component.ts

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

interface Student {
  name: string;
  percentage: number;
```

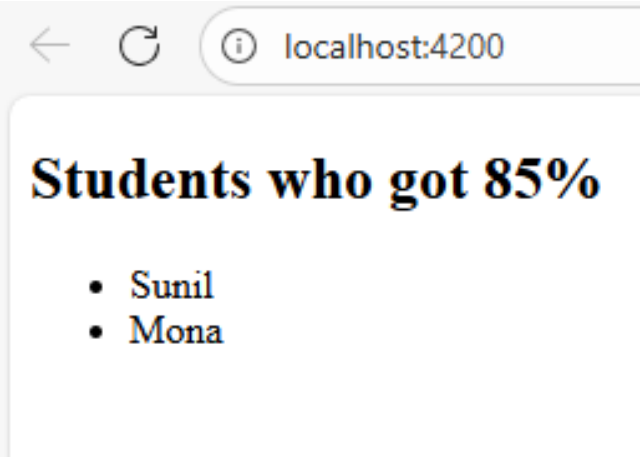
```
}

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  students: Student[] = [
    { name: 'Ram', percentage: 90 },
    { name: 'Sunil', percentage: 85 },
    { name: 'Suresh', percentage: 82 },
    { name: 'Mona', percentage: 85 },
    { name: 'Sona', percentage: 78 },
  ];

  studentsWith85: string[] = [];

  ngOnInit(): void {
    this.studentsWith85 = this.students
      // filter those whose percentage is exactly 85
      .filter(s => s.percentage === 85)
      // then map to just the names
      .map(s => s.name);
  }
}
```

app.component.html

```
<h2>Students who got 85%</h2>
<ul>
```

```
  <li *ngFor="let name of studentsWith85">{{ name }}</li>
</ul>
```

**********output-**********



7) Fetch the details using ng-repeat in AngularJS

```
<!DOCTYPE html>
<html>

<head>
    <script src=
"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
    </script>
    <title>
        Fetching the details using the ng-repeat Directive
```

```
    </title>
    <style>
        body {
            text-align: center;
            font-family: Arial, Helvetica, sans-serif;
        }

        table {
            margin-left: auto;
            margin-right: auto;
        }
    </style>
</head>

<body ng-app="myTable">
    <h1 style="color:green">GeeksforGeeks</h1>
    <h3>
        Fetching the details using the ng-repeat Directive
    </h3>

    <table ng-controller="control" border="2">
        <tr ng-repeat="x in records">
            <td>{{x.Country}}</td>
            <td>{{x.Capital}}</td>
        </tr>
    </table>

    <script>
        var app = angular.module("myTable", []);
        app.controller("control", function ($scope) {
            $scope.records = [
                {
```

```
            "Country": "India",
            "Capital": "Delhi"
        },
        {
            "Country": "America ",
            "Capital": "Washington, D.C. "
        },
        {
            "Country": "Germany",
            "Capital": "Berlin"
        },
        {
            "Country": "Japan",
            "Capital": "Tokyo"
        }
    ]
});
</script>
</body>
</html>
```

**********output-**********

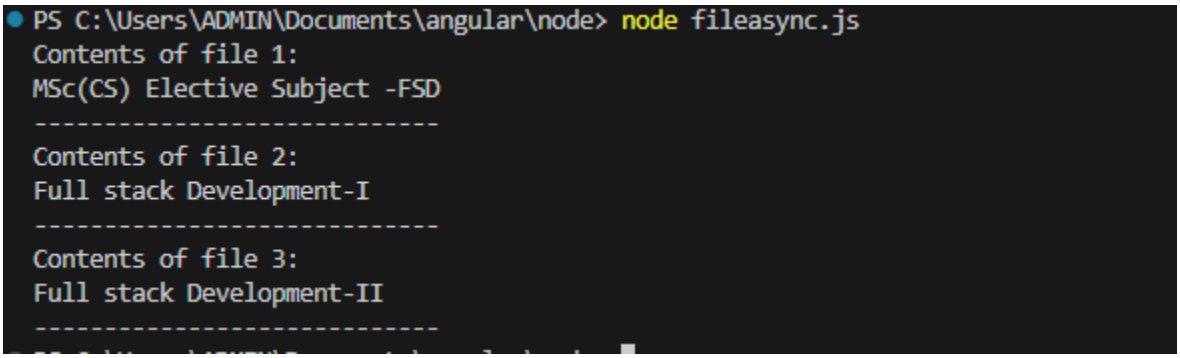
India	Delhi
America	Washington, D.C.
Germany	Berlin
Japan	Tokyo

8) Create a Node.js application that reads data from multiple files asynchronously using promises and async/await

```
// Import the 'fs/promises' module for asynchronous file system operations
const fs = require('fs').promises;


// Asynchronous function to read multiple files
async function readFiles(filePaths) {
  try {
    // Read all files asynchronously using Promise.all
```

```
    const fileContents = await Promise.all(
      filePaths.map(file => fs.readFile(file, 'utf8'))
    );

    // Display content of each file
    fileContents.forEach((content, index) => {
      console.log(`\n📄 Contents of file ${index + 1} (${filePaths[index]}):`);
      console.log(content);
      console.log('----------------------------');
    });

  } catch (error) {
    console.error("❌ Error reading files:", error.message);
  }
}

// Define the file paths (make sure these files exist in your directory)
const files = ['file1.txt', 'file2.txt', 'file3.txt'];

readFiles(files);
```

**********output-**********



9) Implement a simple server using Node.js.

first.js

Extension: nodejs

first.js > http.createServer() callback

```
1   var http= require('http');
2   http.createServer(function(req,res){
3       res.writeHead(200,{'content-type':'text/html'});
4       res.write('Hello ,FYMsc(CS)');
5       res.end();
6   }).listen(8099, function() {
7       console.log('Server is listening on port 8099');
8   });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\admin\Documents\NodeJSPro> node first.js
Server is listening on port 8099
```

10) Develop an Express.js application that defines routes for Create, Update operations on a resource (Employee).

```
app.use(express.json());
```

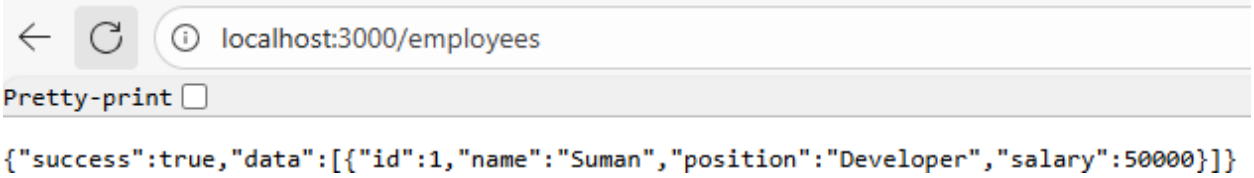
```
let employees = [
    {
        id: 1,
        name: 'Suman',
        position: 'Developer', salary:
        50000
    }
];
let nextEmployeeId = 2;

// CREATE a new employee
// POST /employees
// Body expected: { name: string, position: string, salary: number } app.post('/employees', (req, res)
=> {
    const { name, position, salary } = req.body;

    // Validate input
    if (!name || typeof name !== 'string') {
        return res.status(400).json({ success: false, message: 'Name is required and must be
a string' });
    }
    if (!position || typeof position !== 'string') {
        return res.status(400).json({ success: false, message: 'Position is required and must be a string'
});
    }
    if (salary === undefined || typeof salary !== 'number') {
        return res.status(400).json({ success: false, message: 'Salary is required and must be a
number' });
    }
    const newEmployee = {
        id: nextEmployeeId++, name,
        position, salary
    };
    employees.push(newEmployee);
```

```
    res.status(201).json({ success: true, data: newEmployee });
});

// UPDATE an existing employee
// PUT /employees/:id
// Body can include name, position, salary (one or more) app.put('/employees/:id', (req, res) => {
    const id = parseInt(req.params.id, 10); if
    (isNaN(id)) {
        return res.status(400).json({ success: false, message: 'Invalid employee ID' });
    }

    const employee = employees.find(emp => emp.id === id); if
    (!employee) {
        return res.status(404).json({ success: false, message: 'Employee not found' });
    }

    const { name, position, salary } = req.body;

    // Validate each field if present if (name !==
    undefined) {
        if (typeof name !== 'string') {
            return res.status(400).json({ success: false, message: 'Name must be a string' });
        }
        employee.name = name;
    }
    if (position !== undefined) {
        if (typeof position !== 'string') {
            return res.status(400).json({ success: false, message: 'Position must be a string' });
        }
        employee.position = position;
    }
    if (salary !== undefined) {
        if (typeof salary !== 'number') {
```

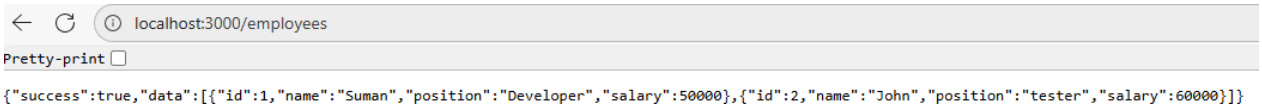
```
        return res.status(400).json({ success: false, message: 'Salary must be a number' });
      }
      employee.salary = salary;
    }

    res.json({ success: true, data: employee });
});

// Optional: GET to view all or one employee
app.get('/employees', (req, res) => {
    res.json({ success: true, data: employees });



// Start server const PORT
= 3000;

app.listen(PORT, () => {
    console.log(`Employee API running on port ${PORT}`);
```

**********output-**********

```
PS C:\Users\ADMIN\Documents\angular\node> node employee.js
Employee API running on port 3000
```



```
PS C:\Users\ADMIN\Documents\angular> Invoke-RestMethod `
>>    -Uri "http://localhost:3000/employees" `
>>    -Method POST `
>>    -Headers @{ "Content-Type" = "application/json" } `
>>    -Body '{"name":"John","position":"tester","salary":60000}'
3;C
success data
------- ----
   True @{id=2; name=John; position=tester; salary=60000}
```



Update request

```
PS C:\Users\ADMIN\Documents\angular> Invoke-RestMethod `
>>    -Uri "http://localhost:3000/employees/1" `
>>    -Method PUT `
>>    -Headers @{ "Content-Type" = "application/json" } `
>>    -Body '{"position":"Senior Developer","salary":75000}'

success data
------- ----
   True @{id=1; name=Suman; position=Senior Developer; salary=75000}
```