KONGU ENGINEERING COLLEGE, PERUNDURAI 638 060
CONTINUOUS ASSESSMENT TEST - III
(Regulations 2020)

Month and Year : June 2023	Roll Number :
Programme : B.E/B.Tech Branch : IT Semester : IV	Date : Time :
Course Code : 20ITT44 Course Name : Web Technology	Duration : 1:30 Hours Max. Marks : 50

PART - A (10×2 = 20 Marks)

ANSWER ALL THE QUESTIONS

1. **Give the features of typescript.** [CO4,K1]

Ans:
- Static Typing
- Modules Support
- Object Oriented Programming Support
- Open Source
- Cross Platform Compatibility
- Supports tools like Emacs, Vim, Atom, WebStorm

2. **How array differs from tuples in typescript. Give an example.** [CO4,K2]

Ans:

Arrays:
- Array is a homogeneous collection of values
- It is a collection of the same dataytpe
- It is allocated with sequential memory blocks

Ex: let StudentDetails: any[ ] = [“Ram”, 1001, “Bangalore”]

Tuples:
- Tuples represents a heterogeneous collection of values
- It is a collection of same or different datatypes
- It is allocated with sequential memory blocks

```
Ex: let person = ["Alice", 25];
console.log(person[0]);    // Output: Alice
console.log(person[1]);
```

3. **Write the Lambda function to perform sum and average of given numbers.** [CO4,K3]

Ans:

```
var add=(x,y)=>x+y;
console.log(add(4,5));
console.log((add(4,5))/2);
```

4. **How do you create group constant using enum for displaying shoe sizes in typescript?** [CO4,K3]

Ans:

```
enum ShoeSize { s=8, m, l} (or)
enum ShoeSize { s=5, m=10, l=15}
```

5.	**What is single page application of angular and how is it accomplished?** Ans: •Single Page Application(SPA) are applications often more dynamic interactions with reduced amount of page refreshes •It is accomplished using AJAX a way to communicate with back-end servers without doing a full page refresh to get data loaded into the application	[CO4,K2]
6.	**What do you understand by templates in angular? Give its two types with syntax.** Ans: Templates separates view layer from the rest of the application and can able to change view layer without affecting other layers. *Inline Template* template: ‘<h1> Welcome </h1>’ *External Template* templateUrl:’./app.component.html’	[CO4,K1]
7.	**Specify the use of pipe for formatting date in angular.** Ans: Pipes are used to format the data before displaying it to the user. • Date – {{todaydate | date:'d/M/y'}} – {{todaydate | date:'shortTime'}}	[CO4,K3]
8.	**Differentiate between template driven form and reactive form in angular?** Ans: **– Template driven Forms** • Fields are controlled as a whole • Used to create small to medium sized forms • Entire form validation is done in HTML templates • Ex. <form (ngSubmit)=”onSubmit()” #courseForm=”ngForm”> **– Model driven Forms or Reactive Forms** • Individual fields can be controlled separately • Used to create large sized forms • Entire form validation is done in Component class using FormBuilder and Validators classes • Ex. <form [formGroup]=”registerForm”>	[CO5,K2]
9.	**Mention the use and role of service in displaying current date and time using angular program.** Ans:	[CO5,K3]

	• Service is used to abstract shared code and functionality throughout the application • Service is used to provide modularity, reusability and adaptability to the application • In dispalying current date and time, service class gets and returns the date and the component class sets it to its instance thus dependancy injection is achieved.		
10.	**How can asynchronous HTTP requests be handled in angular?** Ans: • Generally, HTTP requests are asynchronous meaning that each HTTP request and response are processed independently without waiting for the previous request response • This functionality is achieved in angular code by using Observables which is a reactive programming to retrieve multiple data asynchronously		[CO5,K2]
	PART – B (3 × 10 = 30 Marks) ANSWER ANY THREE QUESTIONS		

11.	**a) Explain about three types of function using default, optional and rest parameters in typescript with an example.**
	Function organizes the program into logical blocks of code and is reusable to perform a specific task that contains function's name, return type and parameters.
	Optional parameters:
	Optional parameters are used when all the values to all the parameters need not be passed mandatorily. A parameter can be made optional by appending a question mark to its name. It should be the last argument in a function.
	Ex. function add(a: number, b?: number, c?: number) {
	  return a + b + (c||1);
	}
	console.log(add(5))
	Default parameters:
	Parameters in a function definition can be assigned with default values and can be explicitly passed with other values during function invocation.
	Ex. function add(a: number, b: number=5) {
	  return a + (b*1);
	}
	console.log(add(2,8))
	Rest parameters:
	Rest parameters are variable-length arguments with same type and do not restrict the number of arguments passed to a rest parameter.
	Ex. function add(a: number, b: number, ...c: number[]) {
		(6)	[CO4,K2]

```
 return a + b + c.length;
}


console.log(add(10,10,10,10,10));
```

b) Outline the three types of access modifiers in typescript with an example. (4)

Access modifiers are used to provide certain restriction of accessing the properties and methods outside the class.

a. public

Typescript properties and methods have default access type as public. Properties declared using public can be accessible outside the class.

b. private

Properties and methods declared with private access can be accessible within the class and cannot be accessible outside the class.

c. protected

Properties and methods declared with protected access can be accessible within the class and inside the inherited classes.

Ex.

```
class Iloan{
 protected interest:number
 //private rebate:number
 rebate:number
  constructor(i:number, r:number){
    this.interest=i
    this.rebate= r
  }
}
class AgriLoan extends Iloan{
  disp():void{
   console.log("Interest amount:"+this.interest)
   console.log("Rebate amount:"+this.rebate)
  }
}
```

```
var obj=new AgriLoan(3,5)
obj.disp()
```

12. **a) Mention the use of static variables and methods in typescript for displaying number of years of loan duration.** (4) [CO4,K3]

In typescript, static variables and methods belong to the class itself rather than instances of the class. It can be accessed without creating an instance of the class and are useful for storing and accessing shared data or functionality across multiple instances of the class.

Ex.

```
class svm{
  static loan_duration=5;

  static disp(){
    console.log(svm. loan_duration--);
  }
}
svm.disp();
svm.disp();
svm.disp();
svm.disp();
```

(6)

b) Write the typescript program to manage the loan information like interest and rebate using class and interface illustrating single and multiple inheritance concepts.

Single inheritance inheriting class:

```
class loan{
  protected interest:number=78;
}
class Agriloan extends loan{
  disp(){
   console.log("inherited class:"+this.interest);
 }
}
```

	const a=new Agriloan();
	a.disp();
	
	Multiple inheritance inheriting interface:
	interface iloan{
	  i
	}
	interface rloan{
	  r
	}
	class Agriloan implements iloan, rloan{
	 i
	 r
	constructor(i,r){
	   this.i=i
	   this.r=r
	 }
	}
	const a=new Agriloan(34,78);
	console.log("interest:"+a.i+" rebate: "+a.r)		

13.	**a) Recall the different types of data binding in angular for product details.** **One-way Data Binding** For updating from component to view, • *Property Binding* Ex. <input [value] = "product.productName"> • *Attribute Binding* - Attribute binding can be used to set the value of an attribute directly. Ex. <td colspan = "{{ 2+3 }}">Product details</td> • *Style Binding* - [style.styleproperty] Ex. <button [style.color] = "isValid ? 'blue' : 'red' ">Order product</button> For updating from view to component, • *Event Binding* Ex. <button (click) ="onSubmit(productQuantity.value,productPrice.value)">Product Order Form</button>	(4)	[CO4,K3]

Two-way Data Binding

For updating from view to component and from component to view,

Ex. <input [(ngModel)] = "product.productName">

```
export class AppComponent{
var productName=”interactivedisplayboard”}
```

(6)

b) Illustrate about the different categories of directives used in angular with appropriate for any suitable application.

• Directives are used to change the behavior of components or elements.

• Directives can be used in the form of HTML attributes.

• Directives are created using classes attached with @Directive decorator which adds metadata to the class.

Types of directives:

1. Components

Components are directives with a template or view.

@Component decorator is actually @Directive with templates

Ex:

component.ts:

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-alert',
  template: `
    <div class="alert">
      <ng-content></ng-content>
    </div>,
  styles: [`
    .alert {
      padding: 10px;
      background-color: yellow;
      border: 1px solid black;
    }
  `]
})
```

	component.html:
	<app-alert>
	  This is a custom alert component.
	</app-alert>
	 2. Structural Directives -*directive-name = expression
	    - *ngIf, - *ngFor, - *ngSwitch
	Ex:
	component.html:
	<ul><li *ngFor=”let course of courses; let i=index”>
	{{i}}-{{course.name}}</li></ul>
	component.ts:
	export class AppComponent{
	   courses: any[]=[
	     {id:1, name:”Typescript”},
	    {id:2, name:”Angular”}
	  ];
	}
	
	3. Attribute Directives - directives changes the appearance / behavior of a component / element
	component.ts:
	export class AppComponent{
	  colorName:String=’red’;
	}
	component.html:
	<p [ngStyle]=”{color:colorName}”> Demo for attribute directive ngStyle </p>		

14.	**a) Write an angular program to create reactive form with performing inbuilt and custom validations.**
	
	1.Create a new Angular component called ReactiveFormComponent:
	  ng generate component ReactiveForm
	
	2.Open the reactive-form.component.ts file and update it with the following code:
	    import { Component } from '@angular/core';
	import { FormBuilder, FormGroup, Validators } from '@angular/forms';
		(6)	[CO5,K3]

```
@Component({
  selector: 'app-reactive-form',
  templateUrl: './reactive-form.component.html',
  styleUrls: ['./reactive-form.component.css']
})
export class ReactiveFormComponent {
  reactiveForm: FormGroup;
  constructor(private formBuilder: FormBuilder) {
    this.reactiveForm = this.formBuilder.group({
      name: ['', [Validators.required, Validators.minLength(3)]],
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(6)]]
    });
  }
  onSubmit() {
    if (this.reactiveForm.valid) {
      console.log('Form submitted successfully!');
      console.log('Form value:', this.reactiveForm.value);
    } else {
      console.log('Form validation failed!');
    }
  }
}
```

3.Open the reactive-form.component.html file and update it with the following code:

```
      <div>
  <h2>Reactive Form</h2>
  <form [formGroup]="reactiveForm" (ngSubmit)="onSubmit()">
    <div>
      <label for="name">Name:</label>
      <input type="text" id="name" formControlName="name">
                                <div *ngIf="reactiveForm.get('name').invalid &&
(reactiveForm.get('name').dirty || reactiveForm.get('name').touched)">
                        <div *ngIf="reactiveForm.get('name').errors.required">Name is
required.</div>
        <div *ngIf="reactiveForm.get('name').errors.minlength">Name should have at
```

```
least 3 characters.</div>
      </div>
    </div>
    <div>
      <label for="email">Email:</label>
      <input type="email" id="email" formControlName="email">
                                        <div*ngIf="reactiveForm.get('email').invalid        &&
(reactiveForm.get('email').dirty || reactiveForm.get('email').touched)">
                    <div   *ngIf="reactiveForm.get('email').errors.required">Email   is
required.</div>
                    <div*ngIf="reactiveForm.get('email').errors.email">Invalid   email
format.</div>
      </div>
    </div>
    <div>
      <label for="password">Password:</label>
      <input type="password" id="password" formControlName="password">
                                    <div*ngIf="reactiveForm.get('password').invalid       &&
(reactiveForm.get('password').dirty || reactiveForm.get('password').touched)">
              <div  *ngIf="reactiveForm.get('password').errors.required">Password  is
required.</div>
        <div *ngIf="reactiveForm.get('password').errors.minlength">Password should
have at least 6 characters.</div>
      </div>
    </div>
    <button type="submit" [disabled]="reactiveForm.invalid">Submit</button>
  </form>
</div>
```

b) Why is routing done in angular and how is routing path set and component loaded based on the routing path? (4)

- Routing in Angular is used to navigate between different views or components within a single-page application (SPA).
- It allows users to access different sections of the application by changing the

URL, without having to reload the entire page.

- Routing helps in creating a seamless and interactive user experience. In Angular, routing is achieved through the Angular Router module. The router maps URL paths to specific components and renders the appropriate component based on the current URL.

app.module.ts:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent },
  { path: '**', component: PageNotFoundComponent } // Wildcard route for handling
unknown paths
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.html:

```
<!-- app.component.html -->
<div>
  <h1>My App</h1>
  <router-outlet></router-outlet>
</div>

<!-- Example navigation menu -->
<ul>
  <li><a routerLink="/home">Home</a></li>
  <li><a routerLink="/about">About</a></li>
  <li><a routerLink="/contact">Contact</a></li>
</ul>
```

Bloom's Taxonomy Level	Remembering (K1)	Understanding (K2)	Applying (K3)	Analysing (K4)	Evaluating (K5)	Creating (K6)
Percentage	10	26.67	63.33	--	--	--