SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

PRACTICAL RECORD NOTE

STUDENT NAME :

REGISTER NUMBER :

CLASS : III BCA

YEAR & SEMESTER : III Year & V Semester

SUBJECT CODE : UCA23D03J

SUBJECT TITLE : WEB DEVELOPMENT USING ANGULAR JS AND MONGODB

OCTOBER 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by

_____

Register No._____ of ***BCA*** *Degree course for*

UCA23D03J – WEB DEVELOPMENT USING ANGULAR JS AND MONGODB

in the Computer lab in SRM Institute of Science and Technology during the academic year 2025-2026.

Staff In-charge **Head of the Department**

Submitted for Semester Practical Examination held on _____.

Internal Examiner *External Examiner*

INDEX

EX NO: 1 (a)	SWAPPING TWO NUMBER USING A DOCUMENT.WRITE METHOD AND PROMPT BOX
Date:	

AIM

To gets the input from the users in prompt boxes and perform swappingtwo variables using document.write method.

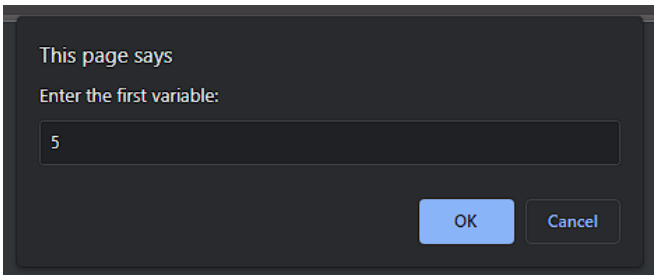
PROCEDURE

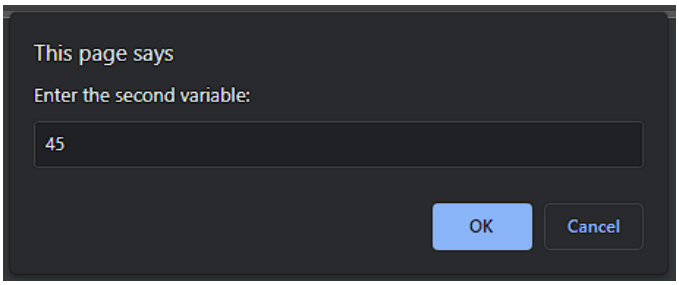
STEP 1: Start The Program.
STEP 2: Create The Html & Script Tags.
STEP 3: Get The Input Variables Using Prompt.
STEP4: Execute The Program In The Browser. STEP 5: Stop.

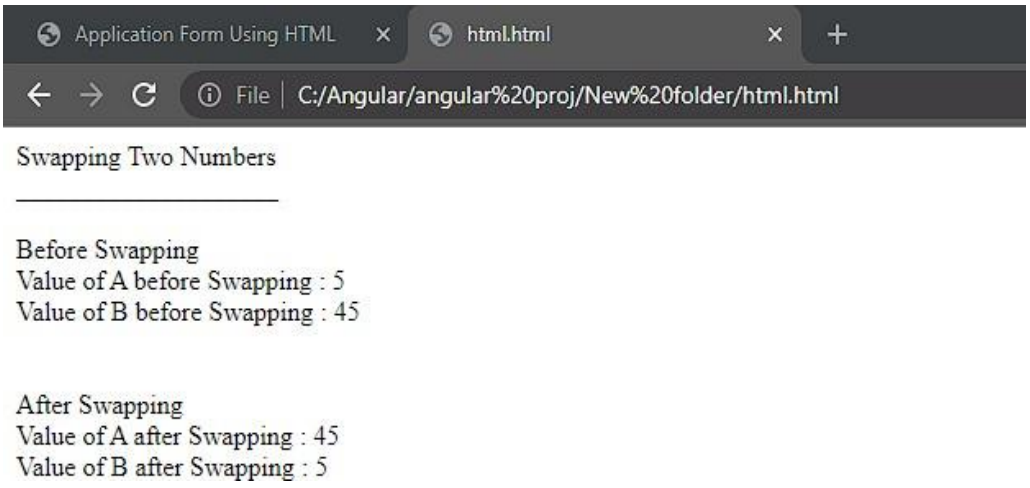
SOURCE CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Swap Two Variables</title>
</head>
<body>
  <script>
    // JavaScript program to swap two variables using document.write method

    var a = parseInt(prompt("Enter the first variable: "));
    var b = parseInt(prompt("Enter the second variable: "));

    document.write("Swapping Two Numbers<br><br>");

    document.write("Before Swapping<br>");
    document.write("Value of A before Swapping: " + a + "<br>");
    document.write("Value of B before Swapping: " + b + "<br>");

    // Swapping without using a temporary variable
    b = a + b;
    a = b - a;
    b = b - a;

    document.write("<br><br>");
    document.write("After Swapping<br>");
    document.write("Value of A after Swapping: " + a + "<br>");
    document.write("Value of B after Swapping: " + b);
  </script>
</body>
</html>
```

OUTPUT







RESULT

Thus, the user input got from prompt box and variables were swapped successfully.

EX NO: 1 (b)	WEB PAGE DESIGN USING INNERHTML()
Date:	

AIM

To design a web page using innerHTML().

PROCEDURE

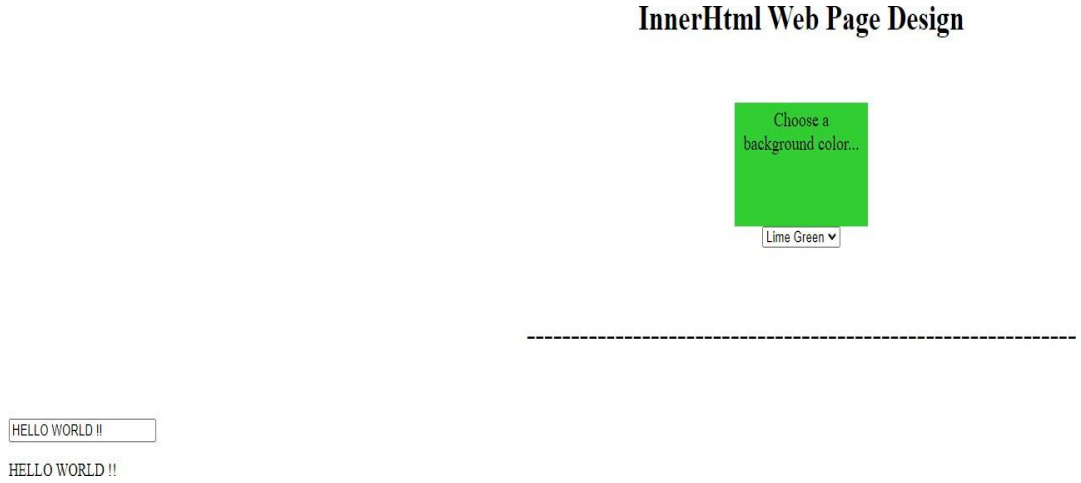
STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Create the functions under the script to perform desired operation.
STEP 4: Execute the program in the browser.
STEP 5: Using Drop-down change the color and enter sample text inside the box.STEP 6: Stop.

SOURCE CODE

```
<html>
    <title>innerHTML</title>
    <center><h1>InnerHtml Web Page Design </h1></center>
    <br><br>
    <script>
        functionchangeColor(){
            var newColor = document.getElementById('colorPicker').value;
            document.getElementById('colorMsg').style.background = newColor;
        }
    </script>
    <center>
        <div id="colorMsg" style="font-size:18px;width:150px;height:100px;padding:5px;"
        align="center">
        Choose a background color...</div>
        <select id="colorPicker" onchange="changeColor()" align="center">
            <option value="transparent">None</option>
            <option value="yellow">Yellow</option>
            <option value="salmon">Salmon</option>
            <option value="lightblue">Light Blue</option>
            <option value="limegreen">Lime Green</option>
        </select>
    </center>
    <br><br>
    <center><h1>.....</h1></center>
    <br><br>
```

```
    <script>
        functionshowMsg(){
            varuserInput = document.getElementById('userInput').value;
            document.getElementById('userMsg').innerHTML = userInput;
        }
    </script>
    <input type="input" maxlength="40" id="userInput" onkeyup="showMsg()"placeholder="Enter text here...">
    <p id="userMsg"></p>
</html>
```

OUTPUT



RESULT

Thus, the page was designed using innerHTML method successfully.

EX NO: 1 (c)	SUM OF TWO NUMBERS USING CONSOLE.LOG
Date:	

AIM

To perform the sum of two numbers and view the output using theconsole.log

PROCEDURE

STEP 1: Start the program.

STEP 2: Create the page using html & script tags.

STEP 3: Perform the desired operation by using proper function inside the script tag.

STEP 4: Execute the program in the browser.

STEP 5: View the output by pressing F12keys (or) right-click on the browser and pressinspect to view the console log.

STEP 6: Stop.

SOURCE CODE

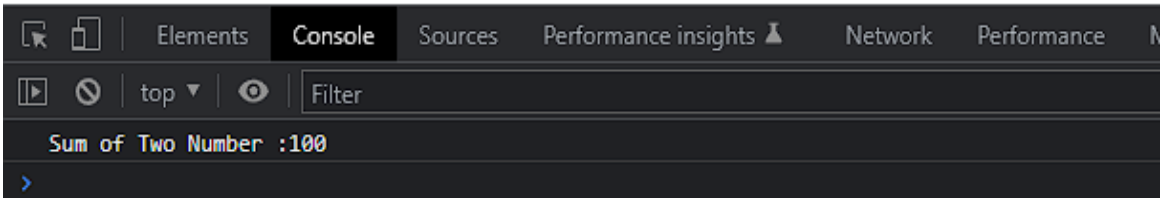
```
<html>
    <body>
        <center><h2>Console.log program </h2></center>
        <label for="fname">Value of A:</label>
        <input type="text" id="fname" name="fname" size="50"><br><br>
        <label for="fname">Value of B:</label>
        <input type="text" id="fname1" name="fname1" size="50"><br><br>
        <input type="button" value="Submit" onclick="sum()">
        <script>
            function sum() {
                let a=parseInt(fname.value); let b=parseInt(fname1.value);c=a+b;
                console.log("Sum of Two Number :"+c);
            }
        </script>
    </body>
</html>
```

OUTPUT

Value of A: 50

Value of B: 50

Submit



RESULT

Thus, the two values were added and output was viewed in console successfully.

EX NO: 2 (a)	ARITHMETIC OPERATION USING SWITCH STATEMENT
Date:	

AIM

To perform the arithmetic operations using a switch statement

PROCEDURE

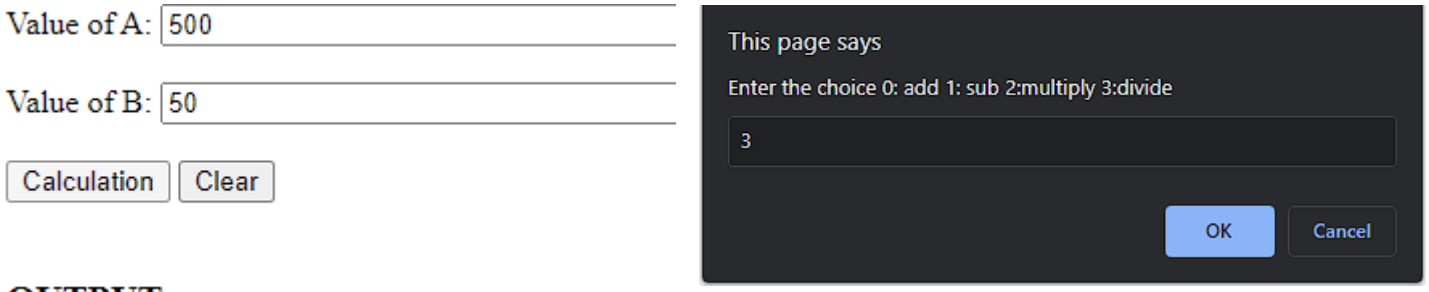
STEP 1: Start the program.
STEP 2: Design the page using html & script tags.
STEP 3: Get the input variables.
STEP 4: Perform the calculation using switch statement.
STEP 5: Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

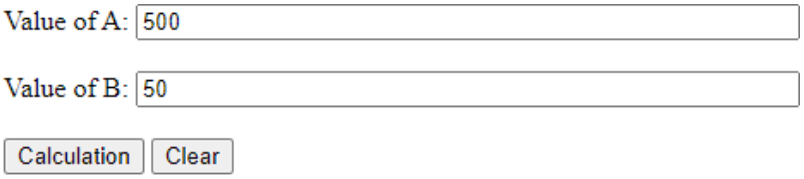
```
<html>
    <body>
    <center><h2>Program using Conditional Statement</h2></center>
    <label for="fname">Value of A:</label>
    <input type="text" id="fname" name="fname" size="50"><br><br>
    <label for="fname">Value of B:</label>
    <input type="text" id="fname1" name="fname1" size="50"><br><br>
    <input type="button" value="Calculation" onclick="switch1()">
    <input type="button" value="Clear" onclick="clear1()">
    <script>
        function switch1()
        {
            let a=parseInt(fname.value); let
            b=parseInt(fname1.value);let
            result;
            if(!a || !b)
            {
                alert("value is missing retry: Enter the A and B value")
            }
            else
```

```
            {
                d=parseInt(prompt("Enter the choice 0: add 1: sub 2:multiply 3:divide"));
                switch (d) {
                    case 0:
                    result=a+b;
                    break;
                    case 1:
                    result=a-b;
                    break;
                    case 2:
                    result=a*b;
                    break;
                    case 3:
                    result=a/b;
                    break;
                }//switch
                document.getElementById("printvalue").innerHTML= "Resultant value : "+result;
            }//else finish
        }//function switch
        function clear1()
        {
            fname.value="";
            fname1.value="";
            fname2.value="";
        }
</script>
<br><br>
<h3> OUTPUT </h3>
<p id="printvalue"></p>
</body>
 </html>
```

OUTPUT



OUTPUT



OUTPUT

Resultant value : 10

RESULT

Thus, the arithmetic operation has been verified successfully.

EX NO: 2 (b)	GREATEST AMONG THREE NUMBERS USING CONDITIONAL STATEMENT
Date:	

AIM

To compare the three numbers and print the greatest number usingconditional statement

PROCEDURE

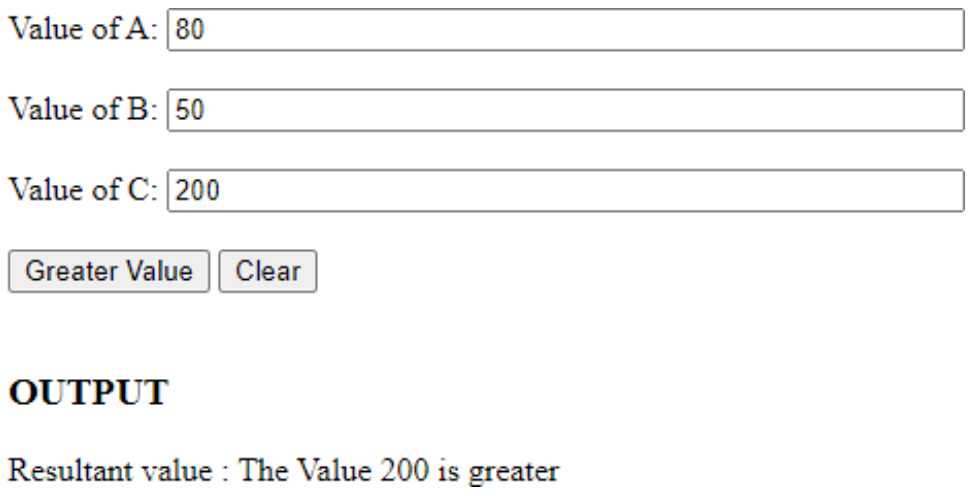
STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Get the input variables.
STEP 4: Perform the comparison process. STEP 5:
Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

```
<html>
  <body>
    <center><h2>Greatest among three numbers using Nested if</h2></center>
    <label for="fname">Value of A:</label>
    <input type="text" id="fname" name="fname" size="50"><br><br>
    <label for="fname">Value of B:</label>
    <input type="text" id="fname1" name="fname1" size="50"><br><br>
    <label for="fname">Value of C:</label>
    <input type="text" id="fname2" name="fname2" size="50"><br><br>
    <input type="button" value="Greater Value" onclick="ife()">
    <input type="button" value="Clear" onclick="clear1()">
    <script>
      function ife(){
        let a=parseInt(fname.value); let b=parseInt(fname1.value); let c=parseInt(fname2.value); let result;
        if(!a || !b || !c) {
          alert("value is missing retry: Enter the A,Band C values")
        }
        else {
          if((a>b)&&(a>c)) {
            result=" The Value "+ a +" is greater";
          }
          else {
            if(b>c) {
              result="The Value "+ b +" is greater";
            }
            else {
              result=" The Value "+ c +" is greater";
            }
          }
          document.getElementById("printvalue").innerHTML= "Resultant value : "+result;
      } }
      function clear1()          {
        fname.value=""; fname1.value=""; fname2.value="";
      }
    </script>
    <br><br>
```

```
    <h3> OUTPUT </h3><p id="printvalue"></p>

  </body>
</html>
```

OUTPUT



RESULT

Thus, the expected output has been verified successfully

EX NO: 3 (a)	FIBONACCI SERIES GENERATION USING WHILE LOOP
Date:	

AIM

To generate Fibonacci series using while loop.

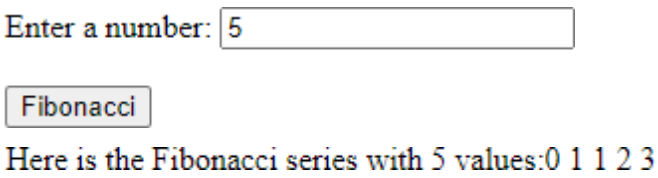
PROCEDURE

STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Get the input variable.
STEP 4: Perform the logic under the function.
STEP 5: Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

```
<html>
   <head>
   <title>JavaScript Prime</title>
   </head>
   <body>
      <center><h1>Fibonacci program using While looping statement</h1>
      </center> Enter a number: <input id = "num"> <br><br>
      <button onclick = "fib()"> Fibonacci </button>
      <script type="text/javascript"> function fib(){
         var var1 = 0, var2 = 1, var3,range_value;
         range_value= parseInt(num.value);
         document.write("Here is the Fibonacci series with 10 values : ");
         while(var1<range_value) {
                  document.write(var1 + " "); var3 = var1+var2;
                  var1 = var2; var2 = var3;
               }
            }
      </script>
   </body>
</html>
```

OUTPUT



RESULT

Thus, the Fibonacci series were generated successfully.

EX NO: 3 (b)	FACTORIAL PROGRAM USING DO..WHILE LOOP
Date:	

AIM

To calculate factorial for a given number using do..while loop

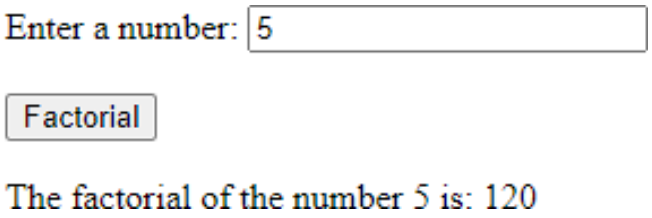
PROCEDURE

STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Get the input variable.
STEP 4: Perform factorial calculation using do..while loop.
STEP 5: Execute the program in the browser.
STEP 6: Stop.

SOURCECODE:

```
<html>
  <body>
    <center><h1>Factorial program using DO while looping statement</h1></center> Enter a number: <input id = "num"> <br><br>
    <button onclick = "fact()"> Factorial </button>
    <p id = "res"></p>
    <script> function fact(){ var i=1, num, f; f = 1;
      num = document.getElementById("num").value;
      do{
        f = f * i; i=i+1; num=num-1;
      }while(num>0); i = i - 1;
      document.getElementById("res").innerHTML = "The factorial of the number " + i + " is: " + f;
    }
    </script>
  </body>
</html>
```

OUTPUT



RESULT

Thus, the factorial was calculated for a given number.

EX NO: 3 (c)	PALINDROME CHECKING USING FOR LOOPING STATEMENT
Date:	

AIM

To check the given string is Palindrome or not using for lopping statement

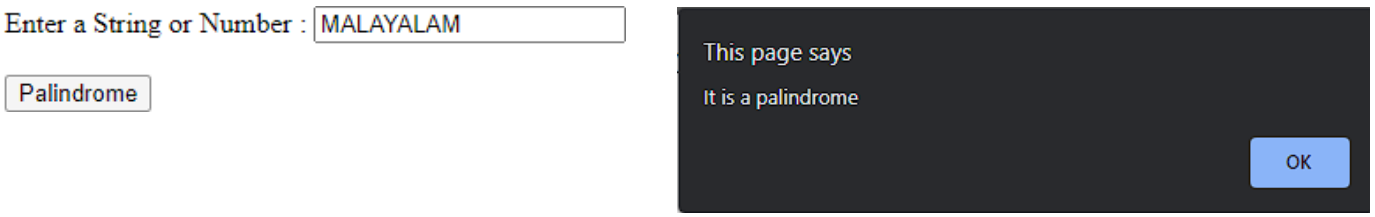
PROCEDURE

STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Get the input variable.
STEP 4: Checking the given string as a palindrome.
STEP 5: Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

```
<html>
   <head>
      <title> JavaScript Palindrome </title>
   </head>
   <body>
         <center>
               <h1>Validate    the Palindrome  numbers or  strings using   For looping   statement</h1>
         </center>
      Enter a String or Number :<input id = "stringinput"><br><br>
      <button onclick = "validatePalin(stringinput.value)"> Palindrome </button>
      <script>
      function validatePalin(str) {
         const string = str;
         var result=0;
         if (str[i] !== str[len - 1 - i]) {
            result+=1;
         }
         if (result==0) {
            alert( 'It is a palindrome');
         }
         else {
            alert( 'It is not a palindrome');
         }
      }
      </script>
   </body>
</html>
```

OUTPUT



RESULT

Thus, the given string is checked whether it a palindrome or not using for loop successfully.

EX NO: 4	PROGRAM USING ARRAY METHODS
Date:	

AIM

To perform array methods for a given input variables.

PROCEDURE

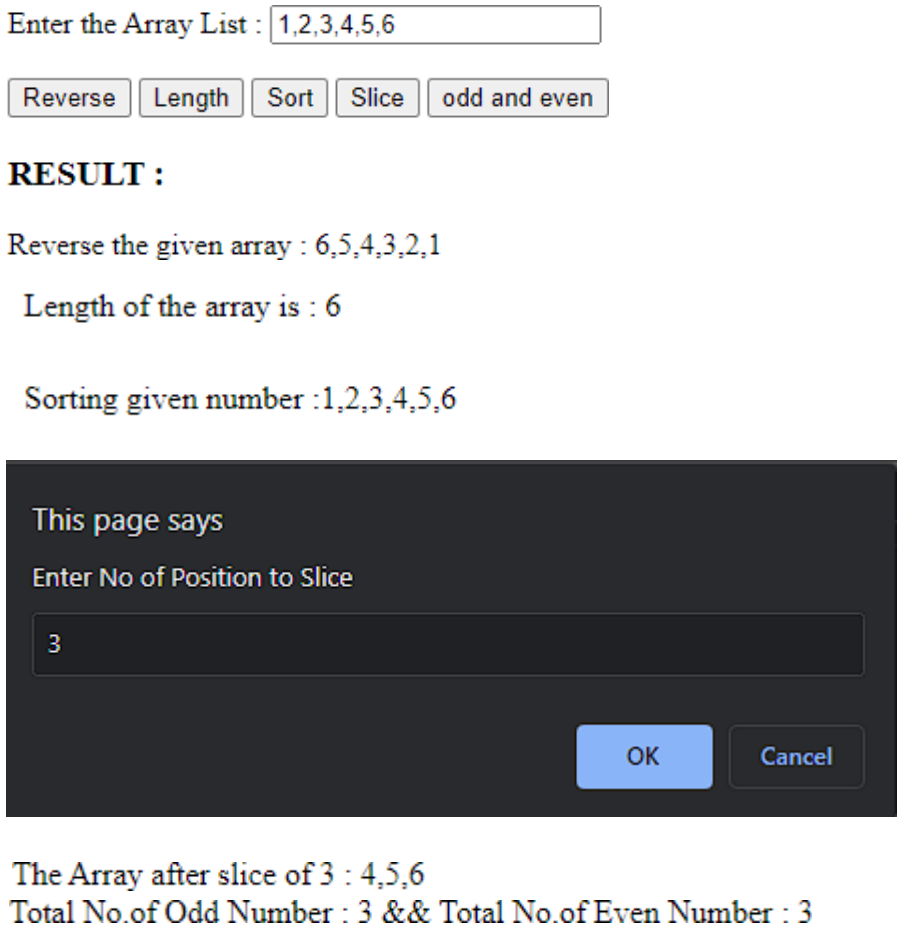
STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Get the input variable.
STEP 4: Create and assign the buttons for the function to perform the required opt.
STEP 5: Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

```
<html>
  <body>
    <center><h1><p> Program Using Array and functions </p></h1></center>
    <label for="arlist">Enter the Array List :</label>
    <input type="text" id="arrvalues" /> <br><br>
    <input type="button" value="Reverse" onclick="reverseof()">
    <input type="button" value="Length" onclick="lengthof()">
    <input type="button" value="Sort" onclick="sortof()">
    <input type="button" value="Slice" onclick="sliceof()">
    <input type="button" value="odd and even" onclick="odds()"> <br>
    <h3>RESULT :</h3><p id="result" name="arlist"/><br>
    <script type="text/javascript">
      varresultarray = new Array();
      function print() {
      vararrayValues = document.getElementById('arrvalues').value;
varsplittedValues = arrayValues.split(',');
      for (let i = 0; i<splittedValues.length; i++) {
      resultarray[i]= splittedValues[i];
      }
      }
      function reverseof(){
        print.call();
        document.getElementById("result").innerHTML =" Reverse the given array :
"+resultarray.reverse();
      }
```

```
        function lengthof(){
            print.call();
            document.getElementById("result").innerHTML ="Length of the array is :
"+resultarray.length;
        }
        function sortof(){
            print.call();
            document.getElementById("result").innerHTML =" Sorting given number :" +
resultarray.sort(function(a, b){return a-b});
        }
        function sliceof(){
            print.call();
            letslice_no=parseInt(prompt("Enter No of Position to Slice"));
            document.getElementById("result").innerHTML ="The Array after slice of "+ slice_no + " :
"+resultarray.slice(slice_no);
        }
        function odds(){
            print.call(); var odd=0;
            var even=0;
            let n=resultarray.length; for(i=0;i<n;i++){
            if((resultarray[i]%2)>0){
                odd=odd+1;
            }
            else{
                even=even+1;
            }
            }
        }
        document.getElementById("result").innerHTML =" Total No.of Odd Number : " + odd + "
&&  Total No.of Even Number : "+even;
    </script>
  </body>
</html>
```

OUTPUT



RESULT

Thus, the expected output has been verified successfully.

EX NO:5	PROGRAM USING ANGULAR JS DIRECTIVES
Date:	

AIM

To implement basic Angular JS directives.

PROCEDURE

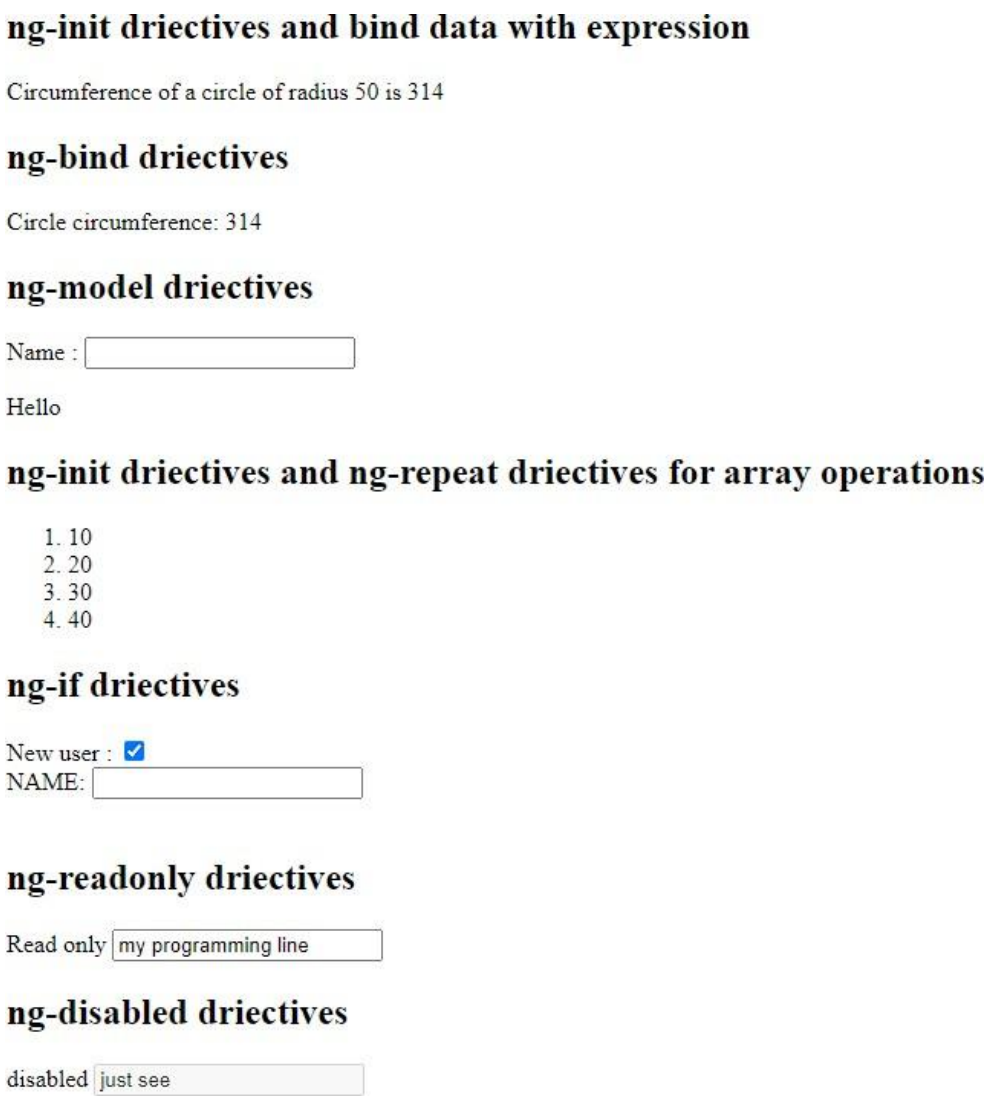
STEP 1: Start the program.
STEP 2: Create the html & script tags.
STEP 3: Initialize the Angular JS directives.
STEP 4: Bind the data using expression.
STEP 5: Save &Execute the program in the browser.
STEP 6: Stop.

SOURCE CODE

```
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Basic Directives</title>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
</head>
<body ng-app="">

  <center>
    <h1>Program based on basic AngularJS Directives</h1>
  </center>

  <!-- ng-init and expression binding -->
  <h2>ng-init directives and bind data with expression</h2>
  <div ng-init="radius=50; PI=3.14">
    Circumference of a circle of radius 50 is {{ 2 * PI * radius }}
    <h2>ng-bind directive</h2>
    Circle circumference:
    <span ng-bind="2 * PI * radius"></span>
    <br>
  </div>

  <!-- ng-model -->
  <h2>ng-model directive</h2>
  <div>
```

```
    <p>Name: <input type="text" ng-model="msg"></p>
    <p>Hello {{ msg }}</p>
  </div>

  <!-- ng-init and ng-repeat -->
  <h2>ng-init and ng-repeat directives for array operations</h2>
  <div ng-init="array=[10,20,30,40]">
    <ol>
      <li ng-repeat="x in array">{{ x }}</li>
    </ol>
  </div>

  <!-- ng-if -->
  <h2>ng-if directive</h2>
  <div ng-init="checked=true">
    New user: <input type="checkbox" ng-model="checked">
    <br>
    NAME: <input type="text" ng-if="checked"><br>
  </div>

  <!-- ng-readonly -->
  <h2>ng-readonly directive</h2>
  Read only:
  <input type="text" ng-readonly="checked" value="my programming line"><br>

  <!-- ng-disabled -->
  <h2>ng-disabled directive</h2>
  Disabled:
  <input type="text" ng-disabled="checked" value="just see"><br>

 </body>
</html>
```

OUTPUT



RESULT

Thus, the expected output has been verified successfully.

EX NO:	CREATE HELLO WORLD USING HOME COMPONENT
Date:	

AIM

To create and demonstrate a simple **Home Component** in AngularJS that displays a "Hello World" message.

PROCEDURE

STEP 1: Open a text editor and create a new file named homeComponent.html**.**

STEP 2 : Add the AngularJS library using a `<script>` tag (CDN link).

STEP 3 : Define the AngularJS application using `angular.module()`.

STEP 4 :Create a Home Component using `app.component("home", {...})`.

STEP 5 :Write the template code to display the text `"Hello World"`.

STEP 6 :Save the file and open it in a web browser.

STEP 7: Observe the output.

SOURCE CODE

```
<html lang="en" ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Home Component</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

  <!-- Using the Home component -->
  <home></home>

  <script>
    // Define the AngularJS app
    var app = angular.module("myApp", []);

    // Create Home component
    app.component("home", {
      template: "<h1>Hello World</h1>"
    });
  </script>

</body>
</html>
```

OUTPUT

ng-init driectives and bind data with expression

Circumference of a circle of radius 50 is 314

ng-bind driectives

Circle circumference: 314

ng-model driectives

Name :

Hello

ng-init driectives and ng-repeat driectives for array operations

1. 10
2. 20
3. 30
4. 40

ng-if driectives

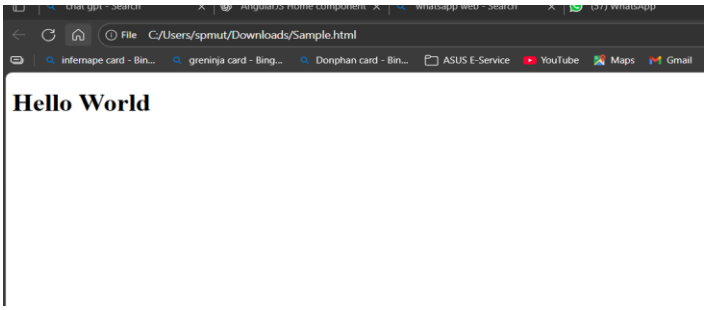
New user :
NAME:

ng-readonly driectives

Read only my programming line

ng-disabled driectives

disabled just see



RESULT

Thus, a simple Home Component was successfully created and executed using AngularJS.

EX NO:	CREATE HOUSING LOCATION COMPONENT
Date:	

AIM

To design and implement a **Housing Location Component** in AngularJS that displays housing details such as name, city, state, and available units.

PROCEDURE

STEP 1: Create a new HTML file (e.g., `housingLocation.html`).

STEP 2 : Load AngularJS library using a CDN `<script>` tag.

STEP 3;Define the AngularJS application using `angular.module()`.

STEP 4: Create a **component** named `"housingLocation"` using `app.component()`.

STEP 5:Inside the component, design a template with housing details.

STEP 6: Add a controller to initialize housing data (`name`, `city`, `state`, `availableUnits`).

STEP 7 :Save the file and open it in a web browser.

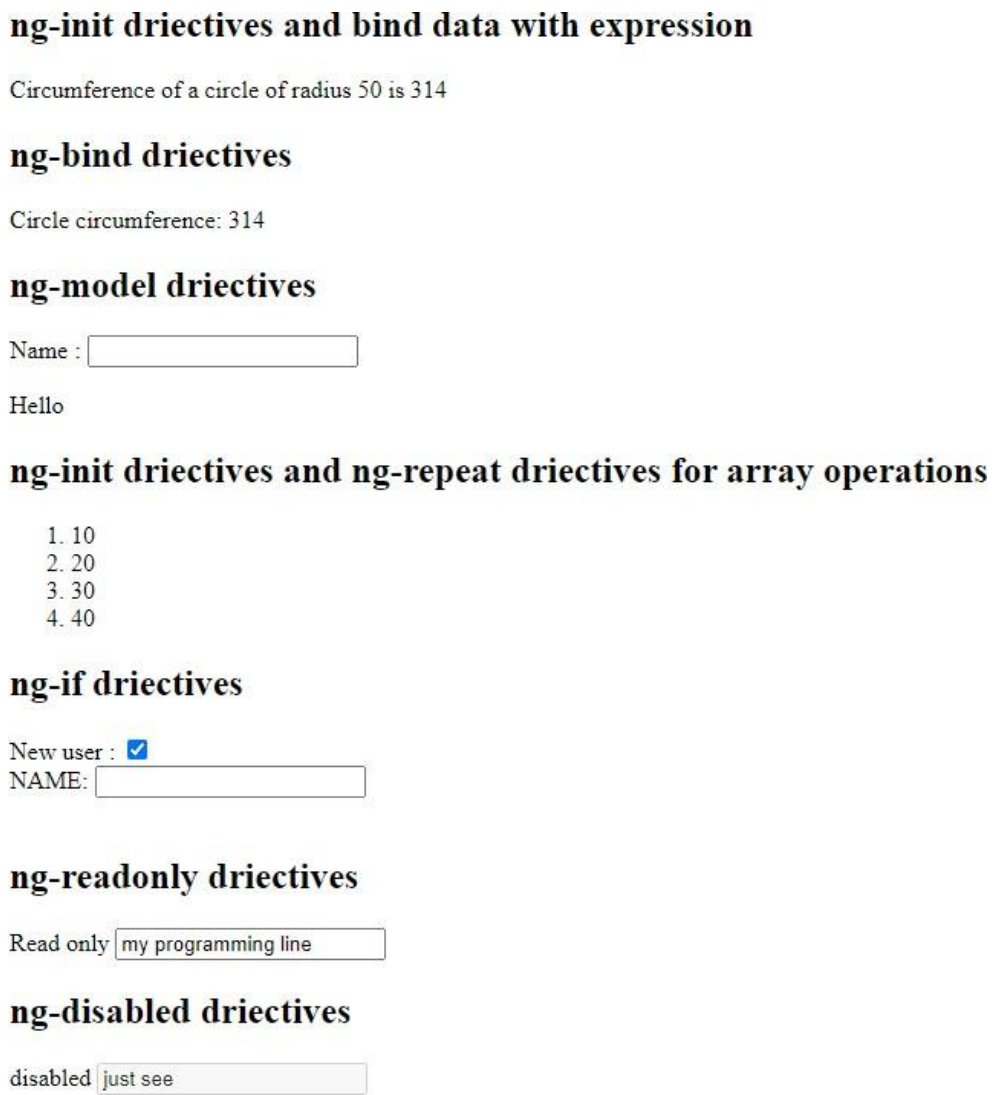
STEP 8 : Observe the displayed housing information.

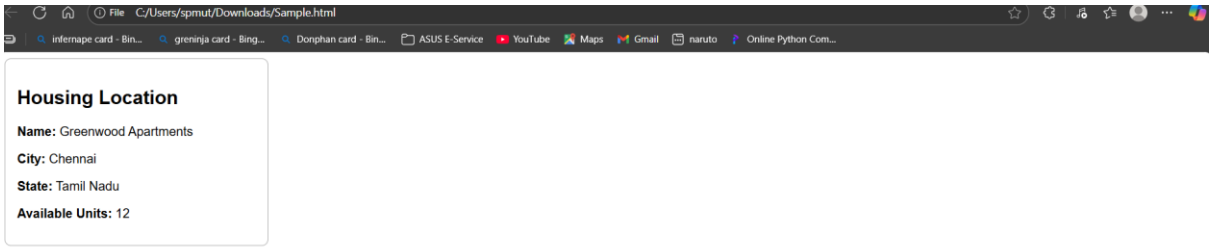
SOURCE CODE

```
<html lang="en" ng-app="myApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Housing Location Component</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>

  <!-- Using the Housing Location component -->
  <housing-location></housing-location>

  <script>
    // Define AngularJS app
    var app = angular.module("myApp", []);

    // Create Housing Location component
    app.component("housingLocation", {
      template: `
        <div style="border:1px solid #aaa; padding:15px; width:300px; border-radius:8px; font-family:Arial;">
```

```
      <h2>Housing Location</h2>
      <p><strong>Name:</strong> {{$ctrl.location.name}}</p>
      <p><strong>City:</strong> {{$ctrl.location.city}}</p>
      <p><strong>State:</strong> {{$ctrl.location.state}}</p>
      <p><strong>Available Units:</strong> {{$ctrl.location.availableUnits}}</p>
    </div>
  `,
  controller: function() {
    this.location = {
      name: "Greenwood Apartments",
      city: "Chennai",
      state: "Tamil Nadu",
      availableUnits: 12
    };
  }
 });
</script>

</body>
</html>
```

OUTPUT





RESULT

Thus, a **Housing Location Component** was successfully created and executed using AngularJS to display housing details dynamically.

EX NO:	ANIMATION
Date:	

AIM

To demonstrate **animations** in AngularJS using the `ngAnimate` module with fade-in and fade-out effects.

PROCEDURE

1. Create a new HTML file (e.g., `animation.html`).
2. Include AngularJS and AngularJS Animate libraries using CDN links.
3. Define an AngularJS app module and include `"ngAnimate"` as a dependency.
4. Create a controller (`MainCtrl`) with a boolean variable (`show`).
5. Add a button that toggles the value of `show`.
6. Use `ng-if` to display the element conditionally.
7. Define CSS classes `.fade.ng-enter` and `.fade.ng-enter-active` to apply animation effects.
8. Save the file and run it in a browser.
9. Observe the animation effect when toggling the button.

SOURCE CODE

```
<html ng-app="myApp">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-animate.js"></script>
  <style>
    .fade.ng-enter { opacity: 0; }
    .fade.ng-enter-active { opacity: 1; transition: 1s; }
  </style>
</head>
<body ng-controller="MainCtrl as main">

  <button ng-click="main.show = !main.show">Toggle</button>

  <div class="fade" ng-if="main.show">
    <h2>Hello World with Animation!</h2>
  </div>

  <script>
    angular.module("myApp", ["ngAnimate"])
      .controller("MainCtrl", function() { this.show = true; });
  </script>
</body>
</html>
```

OUTPUT

EX NO:	PERFORM CRUD OPERATIONS
Date:	

AIM

To Create a student database using MongoDB

PROCEDURE

Step1: Start
Step2:Open Command Prompt
Step3: Run the MongoDB by typing the command mondod
Step 4: Type mongosh to proceed
Step 5: Type the command to get the exact result
Step 6: Stop

Source Code Create Database

test>use student

Display the database

test> show dbs admin 40.00
KiB
config 72.00 KiB
local 72.00 KiB
student 72.00 KiBtest>

Create Collection

student>db.createCollection("stud")
{ ok: 1 }

Insertion

Query db.stud.insert({regno:105,s_name:"jeya",degree:"B.Sc",age:22,CGPA:6.5})
Output

```
{
   acknowledged: true,
   insertedIds: { '0': ObjectId("634a602afcf2121b936e6545") }
}
```

Insert One Record

Query
student> db.stud.insertOne({regno:106,s_name:"Senthil",degree:"B.Sc",age:21,CGPA:7.5})
Output

```
{
   acknowledged: true,
   insertedId: ObjectId("634a60dffcf2121b936e6546")
}
```

Insert Many Record

Query
student> db.stud.insertMany([{regno:107,s_name:"Sankar",degree:"BCA",age:20,CGPA:7.8},{regno:
108,sname:"Hema",degree:"B.Sc",age:20,CGPA:9.8}])
Output

```
{
```

```
  acknowledged: true,insertedIds:
  {
    '0': ObjectId("634a61dffcf2121b936e6547"),
    '1': ObjectId("634a61dffcf2121b936e6548")
  }
}
```

Read

Display all the documents

Query

```
student> db.stud.find()
```

Output

```
[
  {
    _id: ObjectId("634a4ca9d3540c1cf209a048"),regno: 101,
    s_name: 'Kumar',degree:
    'BCA', age: 18,
    CGPA: 7.9
  },
  {
    _id: ObjectId("634a4cf6d3540c1cf209a049"),regno: 102,
    s_name: 'Kirthik',degree:
    'B.Sc', age: 18,
    CGPA: 8.9
  }
]
```

Display all the students in BCA

Query

```
student> db.stud.find({degree:"BCA"})
```

Output

```
[
  {
    _id: ObjectId("634a4ca9d3540c1cf209a048"),regno: 101,
    s_name: 'Kumar',degree:
    'BCA', age: 18,
    CGPA: 7.9
  },
  {
    _id: ObjectId("634a52dcd3540c1cf209a04b"),regno: 104,
    s_name: 'Shanjeev',degree: 'BCA',
    age: 21,
    CGPA: 9
  }
]
```

Updation

Update the student name for the student who is having register number 102

Query `db.stud.updateOne({regno:102},{$set:{s_name:"Muthu"}})`

Output

```
{
    acknowledged: true,insertedId:
    null, matchedCount: 1,
    modifiedCount: 1,
```

```
    upsertedCount: 0
}
```

After Updation

```
student> db.stud.find()
```

Output

```
[
  {
    _id: ObjectId("634a4ca9d3540c1cf209a048"),regno: 101,
    s_name: 'Kumar',degree:
    'BCA', age: 18,
    CGPA: 7.9
  },
  {
    _id: ObjectId("634a4cf6d3540c1cf209a049"),regno: 102,
    s_name: 'Muthu',degree:
    'B.Sc', age: 18,
    CGPA: 8.9
  }
]
```

Update Many

Update the student age

Query

```
student> db.stud.updateMany({age:18},{$set:{age:22}})
```

Output

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
[
  {
    id: ObjectId("634a4ca9d3540c1cf209a048"),regno: 101,
    s_name: 'Kumar',degree:
    'BCA', age: 22,
    CGPA: 7.9
  },
  {
    _id: ObjectId("634a4cf6d3540c1cf209a049"),regno: 102,
    s_name: 'Muthu',degree:
    'B.Sc', age: 22,
    CGPA: 8.9
  }
]
```

Deletion

Delete One Record

Query

```
student> db.stud.deleteOne({s_name:"Kumar"})
```

Output

```
{ acknowledged: true, deletedCount: 1 }
```

Delete Many Record

```
student> db.stud.deleteMany({age:20})

{ acknowledged: true, deletedCount: 2 }
```

Record after deletion

```
student> db.stud.find()
[
  {
    _id: ObjectId("634a4cf6d3540c1cf209a049"),regno: 102,
    s_name: 'Muthu',degree:
    'B.Sc', age: 22,
    CGPA: 8.9
  },
  {
    _id: ObjectId("634a52dcd3540c1cf209a04a"),regno: 103,
    s_name: 'Kirthik',degree:
    'B.Sc', age: 22,
    CGPA: 8.9
  },
  {
    _id: ObjectId("634a52dcd3540c1cf209a04b"),
    regno: 104, s_name: 'Shanjeev',
    degree: 'BCA',
    age: 21,
    CGPA: 9
  },
  {
    _id: ObjectId("634a60dffcf2121b936e6546"),regno: 106,
    sname: 'Senthil',degree:
    'B.Sc', age: 21,
    CGPA: 7.5
  }
]
```

RESULT

Thus CRUD Operation was performed sucessfully

EX NO:	AGGREGATION METHOD
Date:	

Aim

To perform aggregation operations on a student collection in MongoDB to analyze data using operators such as $match, $group, $sort, and $project.

Procedure

STEP 1: Start the MongoDB server and connect to the database.

STEP 2: Create a collection named **students** and insert the given student records using insertMany().

STEP 3:Use the aggregate() function to perform different queries:

- Find average CGPA of all students.
- Find average CGPA grouped by degree.
- Filter students based on CGPA.
- Sort students by CGPA.
- Find youngest student in each degree.
- Display only specific fields (Projection).

2. Observe the output and verify the correctness of aggregation operations.

Coding and Output

1. Insert Records

```
db.students.insertMany([
  { regno: 101, s_name: "Kumar", degree: "BCA", age: 18, CGPA: 7.9 },
  { regno: 102, s_name: "Kirthik", degree: "B.Sc", age: 18, CGPA: 8.9 },
  { regno: 103, s_name: "Priya", degree: "B.Com", age: 19, CGPA: 7.5 },
  { regno: 104, s_name: "Arun", degree: "BCA", age: 20, CGPA: 8.1 },
  { regno: 105, s_name: "Sneha", degree: "B.Sc", age: 18, CGPA: 9.0 },
  { regno: 106, s_name: "Vignesh", degree: "BBA", age: 21, CGPA: 6.9 },
  { regno: 107, s_name: "Lakshmi", degree: "BCA", age: 19, CGPA: 8.7 },
  { regno: 108, s_name: "Rahul", degree: "B.Sc", age: 20, CGPA: 7.8 },
  { regno: 109, s_name: "Divya", degree: "B.Com", age: 18, CGPA: 8.3 },
  { regno: 110, s_name: "Manoj", degree: "BCA", age: 21, CGPA: 7.2 }
])
```

Find Average CGPA of All Students

```
db.students.aggregate([
  { $group: { _id: null, avgCGPA: { $avg: "$CGPA" } } }
])
```

OUTPUT

```
{ "_id": null, "avgCGPA": 8.14 }
```

Find Average CGPA Grouped by Degree

```
db.students.aggregate([
  {
    $group: {
      _id: "$degree",
      avgCGPA: { $avg: "$CGPA" },
```

```
    maxCGPA: { $max: "$CGPA" },
    minCGPA: { $min: "$CGPA" },
    count: { $sum: 1 }
   }
  }
])
```

Output

```
{ "_id": "BCA", "avgCGPA": 7.975, "maxCGPA": 8.7, "minCGPA": 7.2, "count": 4 }
{ "_id": "B.Sc", "avgCGPA": 8.566, "maxCGPA": 9.0, "minCGPA": 7.8, "count": 3 }
{ "_id": "B.Com", "avgCGPA": 7.9, "maxCGPA": 8.3, "minCGPA": 7.5, "count": 2 }
{ "_id": "BBA", "avgCGPA": 6.9, "maxCGPA": 6.9, "minCGPA": 6.9, "count": 1 }
```

Find Students with CGPA > 8.5

```
db.students.aggregate([
  { $match: { CGPA: { $gt: 8.5 } } }
])
```

OUTPUT

```
{ "regno": 102, "s_name": "Kirthik", "degree": "B.Sc", "age": 18, "CGPA": 8.9 }
{ "regno": 105, "s_name": "Sneha", "degree": "B.Sc", "age": 18, "CGPA": 9.0 }
{ "regno": 107, "s_name": "Lakshmi", "degree": "BCA", "age": 19, "CGPA": 8.7 }
```

Sort Students by CGPA (Descending)

```
db.students.aggregate([
  { $sort: { CGPA: -1 } }
])
```

Output

```
{ "regno": 105, "s_name": "Sneha", "degree": "B.Sc", "age": 18, "CGPA": 9.0 }
{ "regno": 102, "s_name": "Kirthik", "degree": "B.Sc", "age": 18, "CGPA": 8.9 }
{ "regno": 107, "s_name": "Lakshmi", "degree": "BCA", "age": 19, "CGPA": 8.7 }
```

Find Youngest Student in Each Degree

```
db.students.aggregate([
  { $group: { _id: "$degree", youngest: { $min: "$age" } } }
])
```

OUTPUT

```
{ "_id": "BCA", "youngest": 18 }
{ "_id": "B.Sc", "youngest": 18 }
{ "_id": "B.Com", "youngest": 18 }
{ "_id": "BBA", "youngest": 21 }
```

Project Only Name and CGPA

```
db.students.aggregate([
  { $project: { _id: 0, Name: "$s_name", CGPA: 1 } }
])
```

```
{ "Name": "Kumar", "CGPA": 7.9 }
{ "Name": "Kirthik", "CGPA": 8.9 }
{ "Name": "Priya", "CGPA": 7.5 }
...
```

EX NO:	IMPLEMENTATION OF INDEXING IN MONGODB
Date:	

AIM

To create and manage indexes in MongoDB collections for improving query performance.

PROCEDURE

STEP 1: Start MongoDB server and connect to the database.
STEP 2:Create a students collection and insert records.
STEP 3:Use the createIndex() method to create different types of indexes.
STEP 4:Run queries before and after creating indexes to observe performance differences.
STEP 5:Display indexes using getIndexes().
STEP 6:Drop an index if needed using dropIndex().

SOURCE CODE

Insert documents

```
db.students.insertMany([
  { regno: 101, s_name: "Kumar", degree: "BCA", age: 18, CGPA: 7.9 },
  { regno: 102, s_name: "Kirthik", degree: "B.Sc", age: 18, CGPA: 8.9 },
  { regno: 103, s_name: "Priya", degree: "B.Com", age: 19, CGPA: 7.5 },
  { regno: 104, s_name: "Arun", degree: "BCA", age: 20, CGPA: 8.1 },
  { regno: 105, s_name: "Sneha", degree: "B.Sc", age: 18, CGPA: 9.0 },
  { regno: 106, s_name: "Vignesh", degree: "BBA", age: 21, CGPA: 6.9 },
  { regno: 107, s_name: "Lakshmi", degree: "BCA", age: 19, CGPA: 8.7 },
  { regno: 108, s_name: "Rahul", degree: "B.Sc", age: 20, CGPA: 7.8 },
  { regno: 109, s_name: "Divya", degree: "B.Com", age: 18, CGPA: 8.3 },
  { regno: 110, s_name: "Manoj", degree: "BCA", age: 21, CGPA: 7.2 }
])
```

Create a Single Field Index on s_name

```
db.students.createIndex({ s_name: 1 })
```

OUTPUT

"s_name_1"

Create an Index on CGPA (Descending)

```
db.students.createIndex({ CGPA: -1 })
```

OUTPUT

"CGPA_-1"

Create a Compound Index on degree and age

```
db.students.createIndex({ degree: 1, age: 1 })
```

OUTPUT

"degree_1_age_1"

Create a Unique Index on regno

db.students.createIndex({ regno: 1 }, { unique: true })

OUTPUT

"regno_1"

View All Indexes in Collection

db.students.getIndexes()

OUTPUT

[
  { "v": 2, "key": { "_id": 1 }, "name": "_id_" },
  { "v": 2, "key": { "s_name": 1 }, "name": "s_name_1" },
  { "v": 2, "key": { "CGPA": -1 }, "name": "CGPA_-1" },
  { "v": 2, "key": { "degree": 1, "age": 1 }, "name": "degree_1_age_1" },
  { "v": 2, "key": { "regno": 1 }, "name": "regno_1", "unique": true }
]

Drop an Index

db.students.dropIndex("s_name_1")

OUTPUT

{ "nIndexesWas": 5, "ok": 1 }

Result

Indexing was successfully implemented on the students collection using MongoDB. Various types of indexes such as single field, compound, descending, and unique indexes were created, verified, and dropped.