Web Programming

Lab - 12

JavaScript

Name: Sushen Grover
Reg No: 23BCE1728
Slot: L11+L12+L31+L32
Class No: CH2024250502774
Course Code: BCSE203E
Faculty: Dr. L.M. Jenila Livingston

Question 1 & 2:

1. Write a **JavaScript program** using the **HTML5 Canvas API** to draw a scene that consists of the following **shapes and corresponding drawings**:

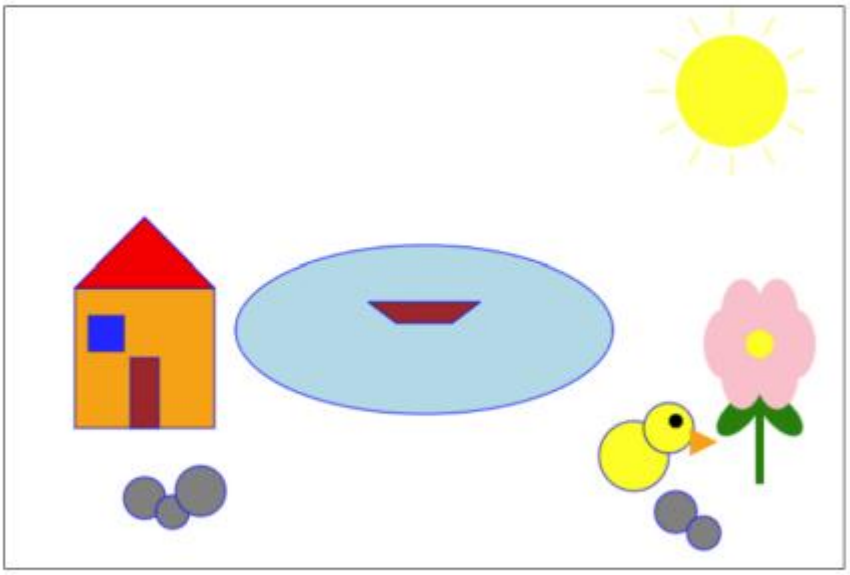
Shape	Drawing Representation
Oval	Pond
Polygon (Quadrilateral with curved edges)	Boat
Two Circles of Different Sizes	Duck (Body & Head)
A Large Circle with Multiple Straight Lines Extending Outward	Sun
A Rectangle with a Triangle on Top	House
An Ellipse with a Vertical Line and Two Curved Shapes	Flower (Stem, Leaves, and Petals)
Multiple Small Circles	Stones

Requirements:

- Use the **Canvas API** functions such as `arc()`, `ellipse()`, `fillRect()`, `lineTo()`, `moveTo()`, and `stroke()`.
- Assign **different colors** to each shape.
- Ensure the **relative positioning** of the elements remains visually structured.

Sample Scene:

Pond Scene using JavaScript Canvas



2. Apply an animation effect to the boat

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Scenery with Boat Animation</title>
</head>
<body>
    <canvas id="canvas" width="800" height="500" style="border:1px solid black;"></canvas>
    <script>
        const canvas = document.getElementById("canvas");
        const ctx = canvas.getContext("2d");

        let boatX = 380;  // Initial X position of the boat
        let boatSpeed = 1.5;  // Speed of the boat
```

```
        function drawScene() {
            ctx.clearRect(0, 0, canvas.width, canvas.height); // Clear canvas before redrawing
```

```
            // Draw House
            ctx.fillStyle = "orange";
            ctx.fillRect(100, 250, 100, 100);
            ctx.fillStyle = "red";
            ctx.beginPath();
            ctx.moveTo(100, 250);
            ctx.lineTo(200, 250);
            ctx.lineTo(150, 200);
            ctx.closePath();
            ctx.fill();
```

```
            ctx.fillStyle = "brown";
            ctx.fillRect(135, 310, 30, 40);
            ctx.fillStyle = "blue";
            ctx.fillRect(110, 270, 20, 20);
```

```
            // Draw Sun
            ctx.fillStyle = "yellow";
            ctx.beginPath();
            ctx.arc(650, 80, 40, 0, Math.PI * 2);
            ctx.fill();
```

```
            // Sun Rays
            ctx.strokeStyle = "yellow";
            for (let i = 0; i < 12; i++) {
                let angle = (Math.PI / 6) * i;
                let x1 = 650 + Math.cos(angle) * 50;
                let y1 = 80 + Math.sin(angle) * 50;
                let x2 = 650 + Math.cos(angle) * 60;
                let y2 = 80 + Math.sin(angle) * 60;
                ctx.beginPath();
                ctx.moveTo(x1, y1);
                ctx.lineTo(x2, y2);
                ctx.stroke();
            }
```

```
            // Draw Pond
            ctx.fillStyle = "lightblue";
            ctx.beginPath();
            ctx.ellipse(400, 350, 120, 60, 0, 0, Math.PI * 2);
            ctx.fill();
            ctx.strokeStyle = "blue";
            ctx.stroke();
```

```
            // Draw Animated Boat
            ctx.fillStyle = "brown";
            ctx.beginPath();
            ctx.moveTo(boatX, 330);
            ctx.lineTo(boatX + 20, 330);
            ctx.lineTo(boatX + 30, 340);
            ctx.lineTo(boatX - 10, 340);
            ctx.closePath();
            ctx.fill();
```

```
            // Draw Bird
            ctx.fillStyle = "yellow";
```

```
            ctx.beginPath();
            ctx.arc(550, 380, 20, 0, Math.PI * 2);
            ctx.fill();
            ctx.beginPath();
            ctx.arc(570, 365, 12, 0, Math.PI * 2);
            ctx.fill();
            ctx.fillStyle = "black";
            ctx.beginPath();
            ctx.arc(575, 360, 3, 0, Math.PI * 2);
            ctx.fill();
            ctx.fillStyle = "orange";
            ctx.beginPath();
            ctx.moveTo(580, 365);
            ctx.lineTo(590, 370);
            ctx.lineTo(580, 375);
            ctx.closePath();
            ctx.fill();
```

```
            // Draw Flower
            ctx.fillStyle = "pink";
            for (let i = 0; i < 6; i++) {
                let angle = (Math.PI / 3) * i;
                let x = 700 + Math.cos(angle) * 20;
                let y = 400 + Math.sin(angle) * 20;
                ctx.beginPath();
                ctx.arc(x, y, 15, 0, Math.PI * 2);
                ctx.fill();
            }
            ctx.fillStyle = "yellow";
            ctx.beginPath();
            ctx.arc(700, 400, 10, 0, Math.PI * 2);
            ctx.fill();
            ctx.fillStyle = "green";
            ctx.fillRect(698, 420, 5, 40);
            ctx.beginPath();
            ctx.arc(690, 430, 10, 0, Math.PI * 2);
            ctx.fill();
            ctx.beginPath();
            ctx.arc(710, 430, 10, 0, Math.PI * 2);
            ctx.fill();
```

```
            // Draw Rocks
            ctx.fillStyle = "gray";
            ctx.beginPath();
            ctx.arc(160, 370, 15, 0, Math.PI * 2);
            ctx.fill();
            ctx.beginPath();
            ctx.arc(180, 380, 12, 0, Math.PI * 2);
            ctx.fill();
            ctx.beginPath();
            ctx.arc(140, 380, 10, 0, Math.PI * 2);
            ctx.fill();
        }
```

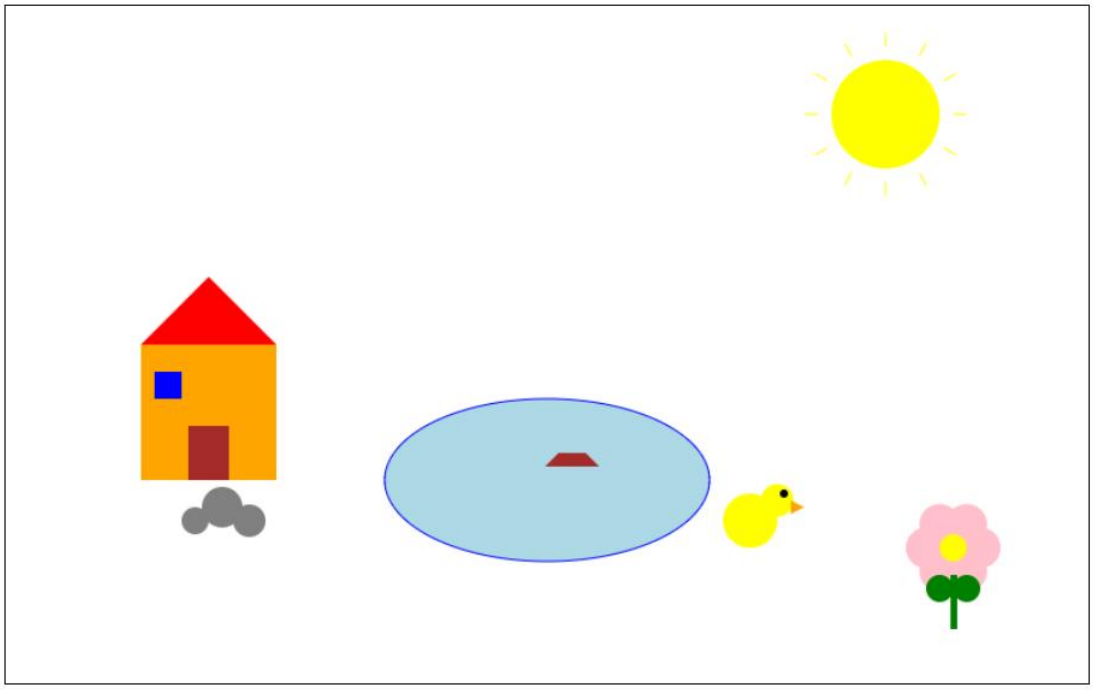
```
        function animateBoat() {
            // Move the boat left and right
            boatX += boatSpeed;
```

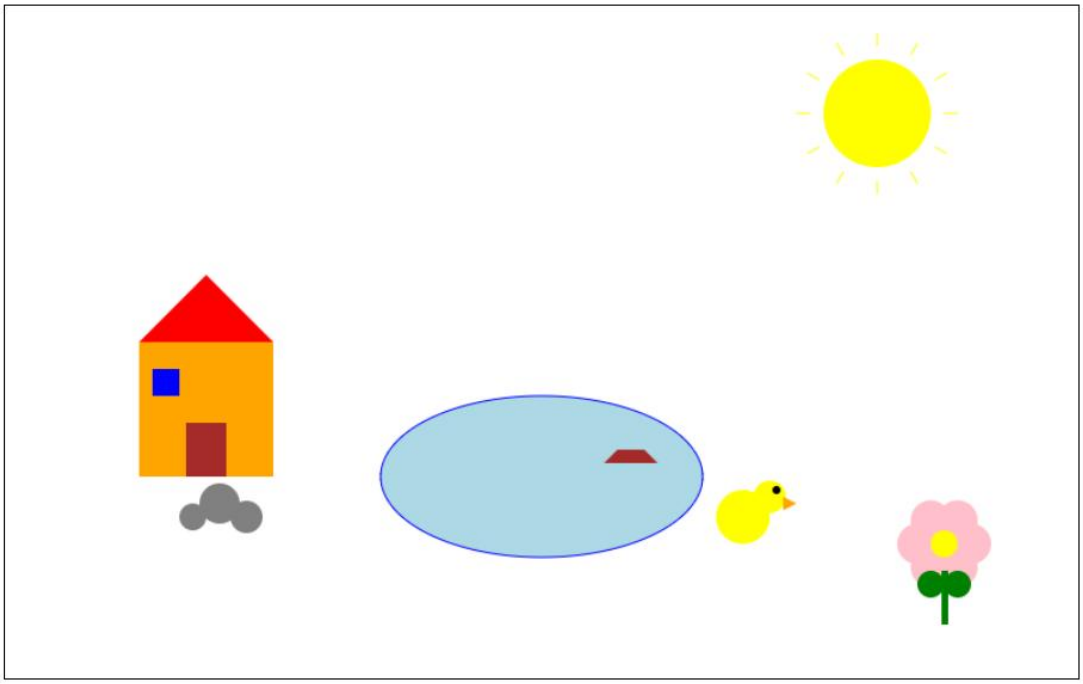
```
            // Change direction if it reaches pond edges
            if (boatX > 460 || boatX < 330) {
                boatSpeed *= -1;
            }
```

```
            drawScene(); // Redraw scene with updated boat position
            requestAnimationFrame(animateBoat); // Recursively animate
        }
```

```
        animateBoat(); // Start animation
    </script>
</body>
</html>
```

Output:





Question 3:

3. Write a JavaScript program that creates a **working analog clock** using the HTML5 Canvas API. The clock should display the **current time dynamically and accurately**, updating every second.

Requirements:

i) Use the Canvas API to draw the clock face, hands, and markings.
ii) The clock must include the following elements:
    a. A circular clock face with a border and a filled background color.
    b. Hour, minute, and second hands that update dynamically based on the current time.
    c. Numerical or tick markings for hours (1 to 12).
    d. A center pivot point for the hands.
iii) Ensure the hands move smoothly and update every second.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analog Clock</title>
</head>
<body onload="startClock()">
    <canvas id="clockCanvas" width="400" height="400"></canvas>
    <script>
        const canvas = document.getElementById("clockCanvas");
        const ctx = canvas.getContext("2d");
        const radius = canvas.width / 2;
        ctx.translate(radius, radius);

        function drawClock() {
            ctx.clearRect(-radius, -radius, canvas.width, canvas.height);
            drawFace();
            drawNumbers();
            drawHands();
            requestAnimationFrame(drawClock);
        }
```

```
        function drawFace() {
            ctx.beginPath();
            ctx.arc(0, 0, radius - 10, 0, Math.PI * 2);
            ctx.fillStyle = "lightblue";
            ctx.fill();
            ctx.lineWidth = 4;
            ctx.stroke();
            ctx.beginPath();
            ctx.arc(0, 0, 5, 0, Math.PI * 2);
            ctx.fillStyle = "black";
            ctx.fill();
        }
```

```
        function drawNumbers() {
            ctx.font = "20px Arial";
            ctx.textAlign = "center";
            ctx.textBaseline = "middle";
            for (let num = 1; num <= 12; num++) {
                let angle = ((num - 3) * Math.PI) / 6; // Correcting the orientation
                let x = Math.cos(angle) * (radius - 30);
                let y = Math.sin(angle) * (radius - 30);
                ctx.fillText(num, x, y);
            }
        }
```

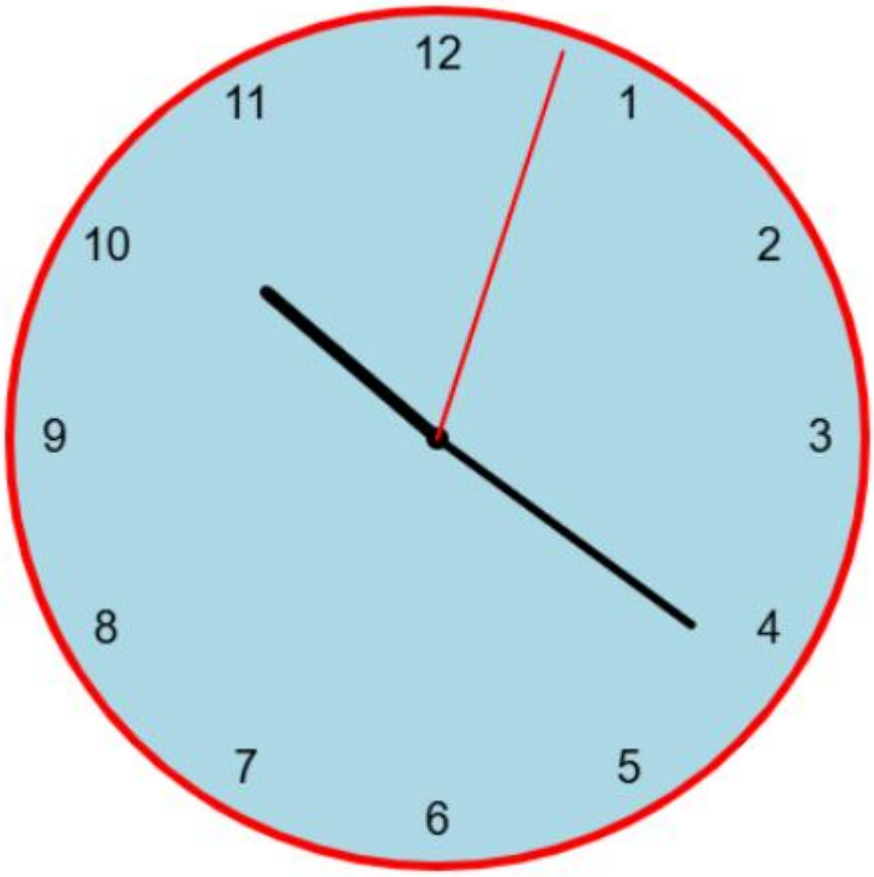
```
        function drawHands() {
            const now = new Date();
            let hours = now.getHours() % 12;
            let minutes = now.getMinutes();
            let seconds = now.getSeconds();
```

```
            drawHand(((hours + minutes / 60) * 30 * Math.PI) / 180, radius * 0.5, 6);
            drawHand(((minutes + seconds / 60) * 6 * Math.PI) / 180, radius * 0.7, 4);
            drawHand((seconds * 6 * Math.PI) / 180, radius * 0.9, 2, "red");
        }
```

```
        function drawHand(angle, length, width, color = "black") {
            ctx.beginPath();
            ctx.lineWidth = width;
            ctx.lineCap = "round";
            ctx.strokeStyle = color;
            ctx.moveTo(0, 0);
            ctx.lineTo(length * Math.cos(angle - Math.PI / 2), length * Math.sin(angle - Math.PI / 2));
            ctx.stroke();
        }
```

```
        function startClock() {
            drawClock();
        }
    </script>
</body>
</html>
```

Output:



Question 4:

4. Write a **JavaScript program** that dynamically generates the charts (**bar chart, line chart, pie chart** and a **donut chart)** using **Plotly.js**.
   Each chart must include:

   a. Labeled X and Y axes (for bar and line charts).
   b. Title for each chart.
   c. Different colors for data points.
   d. Legend (for the pie chart and donut) showing categories.

ii) The chart should be scaled properly to fit within the display area.

Code:

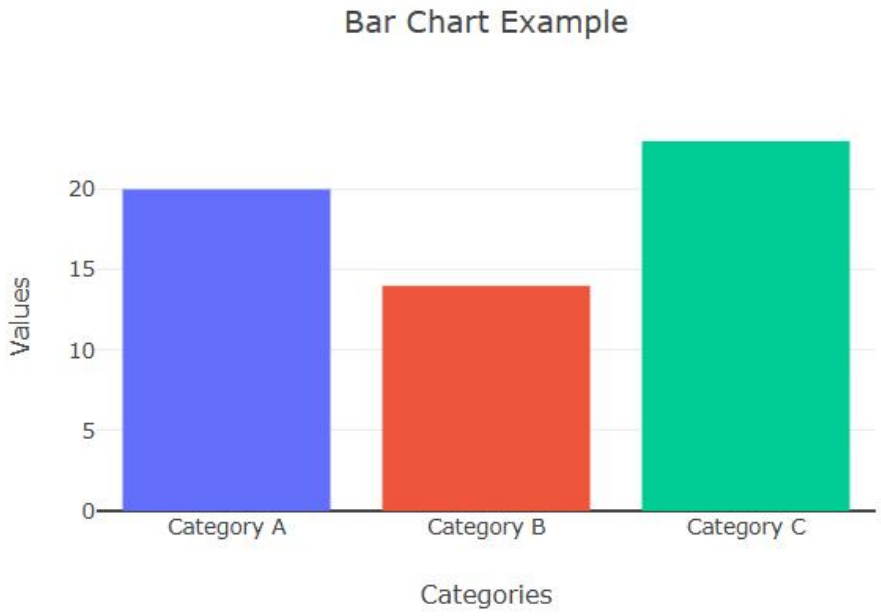
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic Charts with Plotly.js</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
    <!-- Div elements for each chart -->
    <div id="barChart" style="width: 600px; height: 400px;"></div>
    <div id="lineChart" style="width: 600px; height: 400px;"></div>
    <div id="pieChart" style="width: 600px; height: 400px;"></div>
    <div id="donutChart" style="width: 600px; height: 400px;"></div>

    <script>
        // Bar Chart
        const barData = [{
            x: ['Category A', 'Category B', 'Category C'],
            y: [20, 14, 23],
            type: 'bar',
            marker: { color: ['#636EFA', '#EF553B', '#00CC96'] }
        }];
        const barLayout = {
            title: 'Bar Chart Example',
            xaxis: { title: 'Categories' },
            yaxis: { title: 'Values' },
        };
        Plotly.newPlot('barChart', barData, barLayout);
```

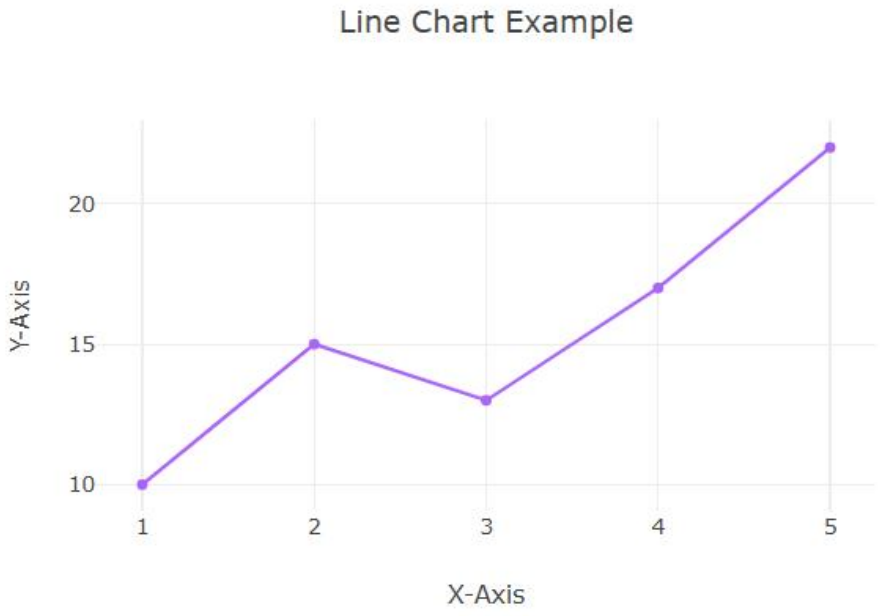
```
        // Line Chart
        const lineData = [{
            x: [1, 2, 3, 4, 5],
            y: [10, 15, 13, 17, 22],
            type: 'scatter',
            mode: 'lines+markers',
            line: { color: '#AB63FA' }
        }];
        const lineLayout = {
            title: 'Line Chart Example',
            xaxis: { title: 'X-Axis' },
            yaxis: { title: 'Y-Axis' },
        };
        Plotly.newPlot('lineChart', lineData, lineLayout);
```

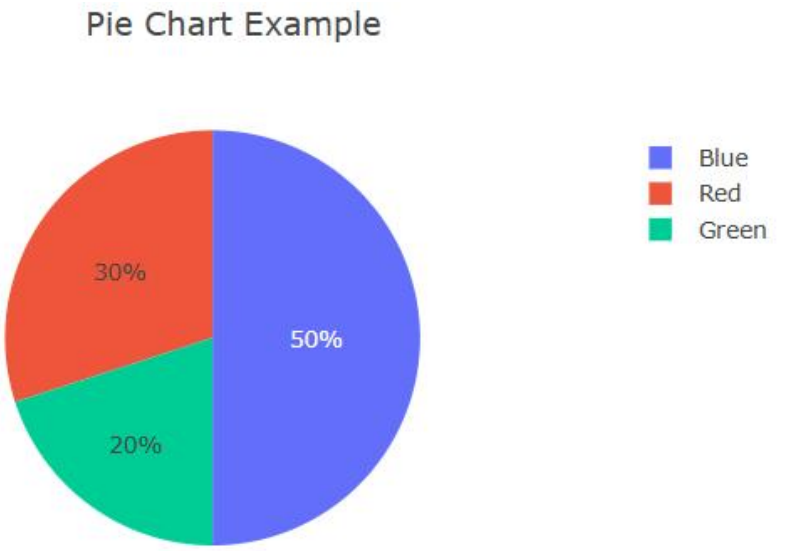
```
        // Pie Chart
        const pieData = [{
            labels: ['Red', 'Blue', 'Green'],
            values: [30, 50, 20],
            type: 'pie',
            marker: {
                colors: ['#EF553B', '#636EFA', '#00CC96']
            }
        }];
        const pieLayout = {
            title: 'Pie Chart Example',
            showlegend: true
        };
        Plotly.newPlot('pieChart', pieData, pieLayout);
```

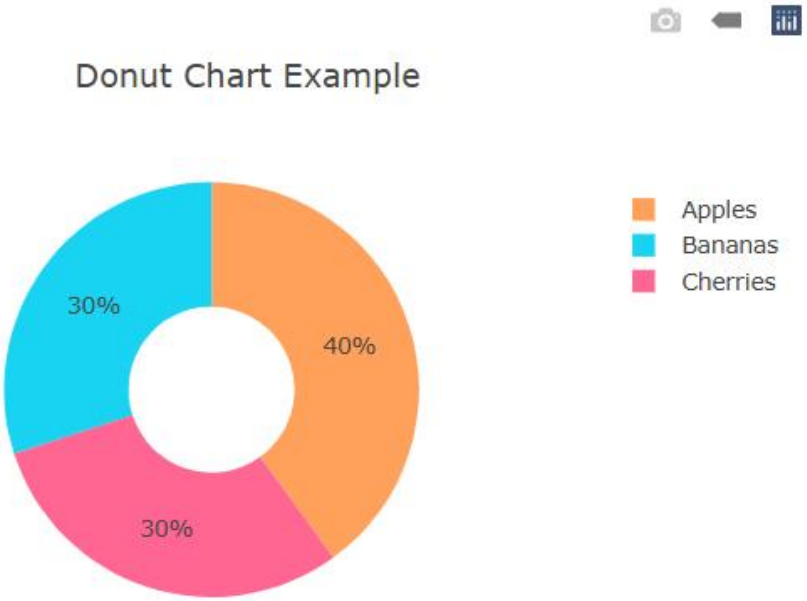
```
        // Donut Chart
        const donutData = [{
            labels: ['Apples', 'Bananas', 'Cherries'],
            values: [40, 30, 30],
            type: 'pie',
            hole: 0.4,
            marker: {
                colors: ['#FFA15A', '#19D3F3', '#FF6692']
            }
        }];
        const donutLayout = {
            title: 'Donut Chart Example',
            showlegend: true
        };
        Plotly.newPlot('donutChart', donutData, donutLayout);
    </script>
</body>
</html>
```

Output:


Bar Chart Example
20
15
10
5
0
Values
Category A
Category B
Category C
Categories


Line Chart Example
20
15
10
Y-Axis
1
2
3
4
5
X-Axis





Question 5:

5. Write a JavaScript program that dynamically creates and manipulates **overlapping elements** using **CSS `z-index`**. The program should allow the user to **change the stacking order** of elements by adjusting their `z-index` values.

    - Create at least three overlapping elements (e.g., `div` boxes or images).
    - Use CSS z-index to control the layering order of these elements.
    - Provide buttons or user input to dynamically adjust the z-index values using JavaScript.
    - Display the current z-index value of each element.

Code:

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Manipulate z-index</title>
    <style>
        /* Styles for the overlapping divs */
        .box {
            width: 150px;
            height: 150px;
            position: absolute;
            border: 2px solid black;
        }

        #box1 {
            background-color: red;
            top: 50px;
            left: 50px;
            z-index: 1;
        }
```

```
        #box2 {
            background-color: blue;
            top: 100px;
            left: 100px;
            z-index: 2;
        }
```

```
        #box3 {
            background-color: green;
            top: 150px;
            left: 150px;
            z-index: 3;
        }
```

```
        /* Container for controls */
        .controls {
            margin-top: 300px;
        }
```

```
        .control-group {
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <!-- Overlapping boxes -->
    <div id="box1" class="box"></div>
    <div id="box2" class="box"></div>
    <div id="box3" class="box"></div>
```

```
    <!-- Controls to adjust z-index -->
    <div class="controls">
        <h3>Adjust Z-Index</h3>

        <!-- Controls for Box 1 -->
        <div class="control-group">
            <label>Box 1 (Red): </label>
            <button onclick="changeZIndex('box1', 'increase')">Increase Z-Index</button>
            <button onclick="changeZIndex('box1', 'decrease')">Decrease Z-Index</button>
            <span id="zIndexBox1">Z-Index: 1</span>
        </div>
```

```
        <!-- Controls for Box 2 -->
        <div class="control-group">
            <label>Box 2 (Blue): </label>
            <button onclick="changeZIndex('box2', 'increase')">Increase Z-Index</button>
            <button onclick="changeZIndex('box2', 'decrease')">Decrease Z-Index</button>
            <span id="zIndexBox2">Z-Index: 2</span>
        </div>
```

```
        <!-- Controls for Box 3 -->
        <div class="control-group">
            <label>Box 3 (Green): </label>
            <button onclick="changeZIndex('box3', 'increase')">Increase Z-Index</button>
            <button onclick="changeZIndex('box3', 'decrease')">Decrease Z-Index</button>
            <span id="zIndexBox3">Z-Index: 3</span>
        </div>
    </div>
```

```
    <!-- JavaScript -->
    <script>
        // Function to change the z-index of an element
        function changeZIndex(boxId, action) {
            // Get the box element
            const box = document.getElementById(boxId);
```

```
            // Get the current z-index value
            let currentZIndex = parseInt(window.getComputedStyle(box).zIndex);
```

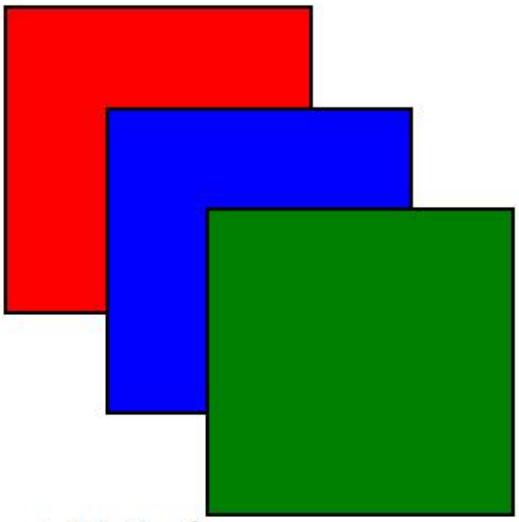
```
            // Adjust the z-index based on the action
            if (action === 'increase') {
                currentZIndex++;
            } else if (action === 'decrease') {
                currentZIndex--;
            }
```

```
            // Update the z-index value of the box
            box.style.zIndex = currentZIndex;
```

```
            // Update the displayed z-index value
            document.getElementById(`zIndex${capitalizeFirstLetter(boxId)}`).innerText = `Z-Index:
${currentZIndex}`;
        }
```

```
        // Helper function to capitalize the first letter of a string
        function capitalizeFirstLetter(string) {
            return string.charAt(0).toUpperCase() + string.slice(1);
        }
    </script>
</body>
</html>
```

Output:

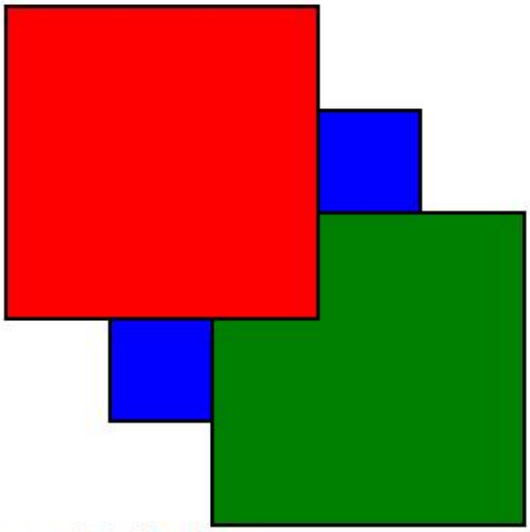


Adjust Z-Index

Box 1 (Red): Increase Z-Index Decrease Z-Index Z-Index: 1

Box 2 (Blue): Increase Z-Index Decrease Z-Index Z-Index: 2

Box 3 (Green): Increase Z-Index Decrease Z-Index Z-Index: 3



Adjust Z-Index

Box 1 (Red): Increase Z-Index Decrease Z-Index Z-Index: 5

Box 2 (Blue): Increase Z-Index Decrease Z-Index Z-Index: 2

Box 3 (Green): Increase Z-Index Decrease Z-Index Z-Index: 3