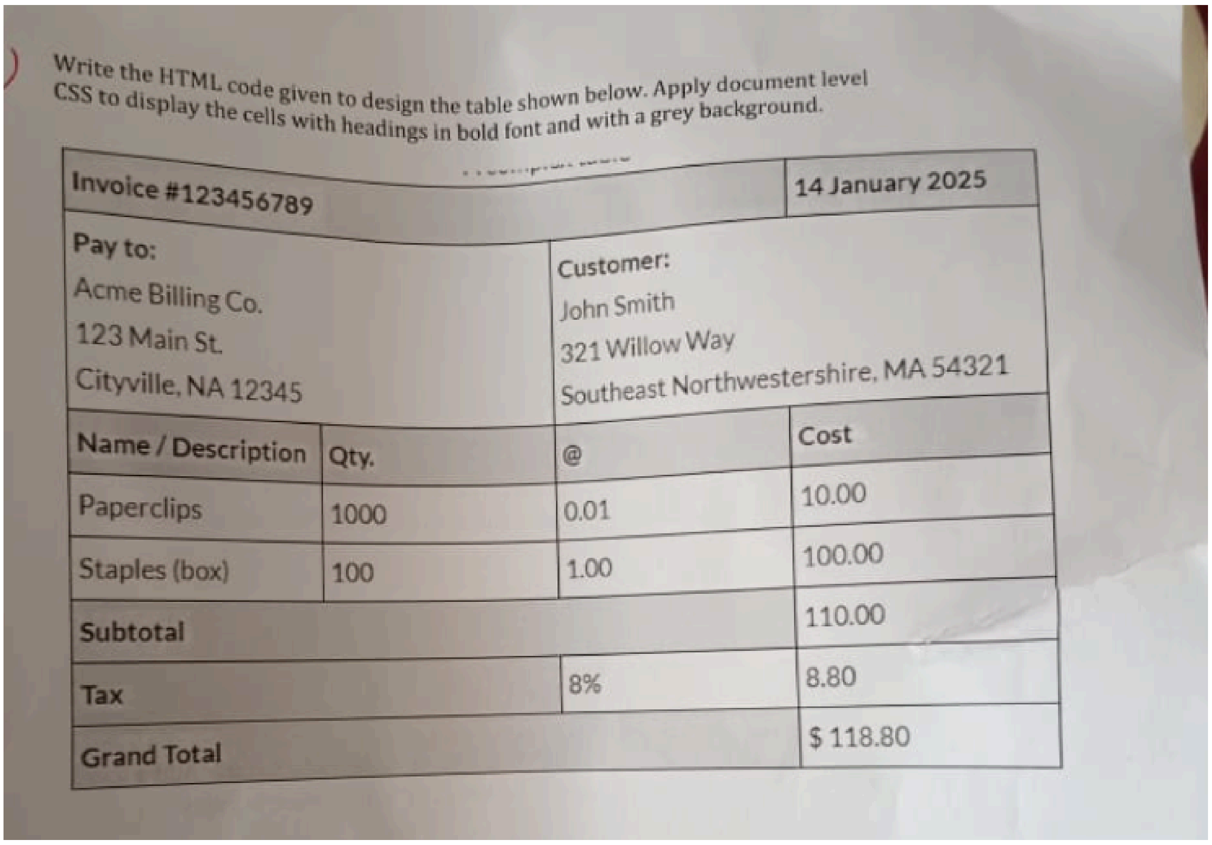
Here's the HTML and CSS code to create the table shown in your image:



Write the HTML code given to design the table shown below. Apply document level CSS to display the cells with headings in bold font and with a grey background.

Invoice #123456789			**14 January 2025**
Pay to: Acme Billing Co. 123 Main St. Cityville, NA 12345		**Customer:** John Smith 321 Willow Way Southeast Northwestershire, MA 54321	
Name / Description	**Qty.**	**@**	**Cost**
Paperclips	1000	0.01	10.00
Staples (box)	100	1.00	100.00
Subtotal			110.00
Tax		8%	8.80
Grand Total			$ 118.80

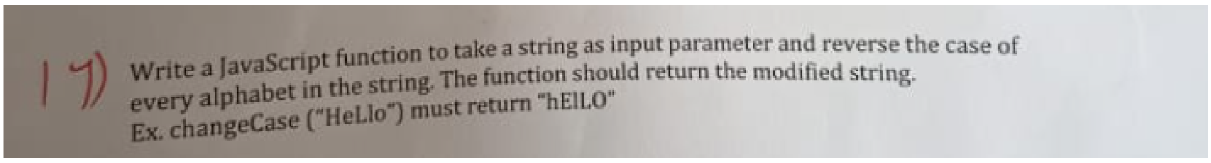
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Invoice</title>
  <style>
    table {
      width: 100%;
      border-collapse: collapse;
      font-family: Arial, sans-serif;
    }

    th, .heading-cell {
      font-weight: bold;
      background-color: #f2f2f2;
    }

    td, th {
      padding: 8px;
      border: 1px solid #ddd;
      text-align: left;
    }
```

```
        .right-align {
            text-align: right;
        }
    </style>
</head>
<body>
    <table>
        <tr>
            <td class="heading-cell" colspan="2">Invoice #123456789</td>
            <td class="heading-cell" colspan="2">14 January 2025</td>
        </tr>
        <tr>
            <td class="heading-cell" colspan="2">Pay to:</td>
            <td class="heading-cell" colspan="2">Customer:</td>
        </tr>
        <tr>
            <td colspan="2">Acme Billing Co.<br>123 Main St.<br>Cityville, NA 12345</td>
            <td colspan="2">John Smith<br>321 Willow Way<br>Southeast Northwesternshire,
MA 54321</td>
        </tr>
        <tr>
            <th>Name / Description</th>
            <th>Qty.</th>
            <th>@</th>
            <th>Cost</th>
        </tr>
        <tr>
            <td>Paper clips</td>
            <td>1000</td>
            <td>0.01</td>
            <td>10.00</td>
        </tr>
        <tr>
            <td>Staples (box)</td>
            <td>100</td>
            <td>1.00</td>
            <td>100.00</td>
        </tr>
        <tr>
            <td class="heading-cell" colspan="3">Subtotal</td>
            <td>110.00</td>
        </tr>
        <tr>
            <td class="heading-cell" colspan="3">Tax</td>
            <td>8.80</td>
        </tr>
        <tr>
```

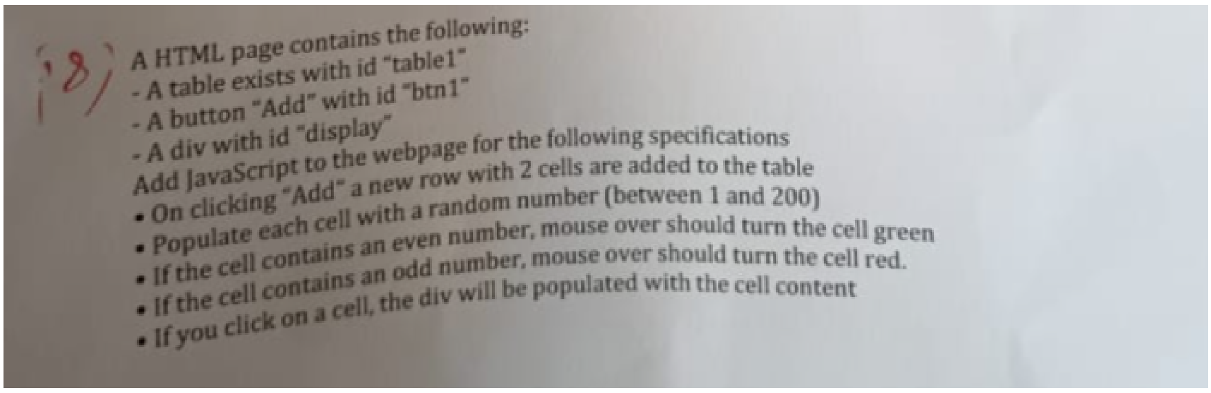
```html
            <td class="heading-cell" colspan="3">Grand Total</td>
            <td>$ 118.80</td>
        </tr>
    </table>
</body>
</html>
```

For Blackbox AI, you could use this prompt:

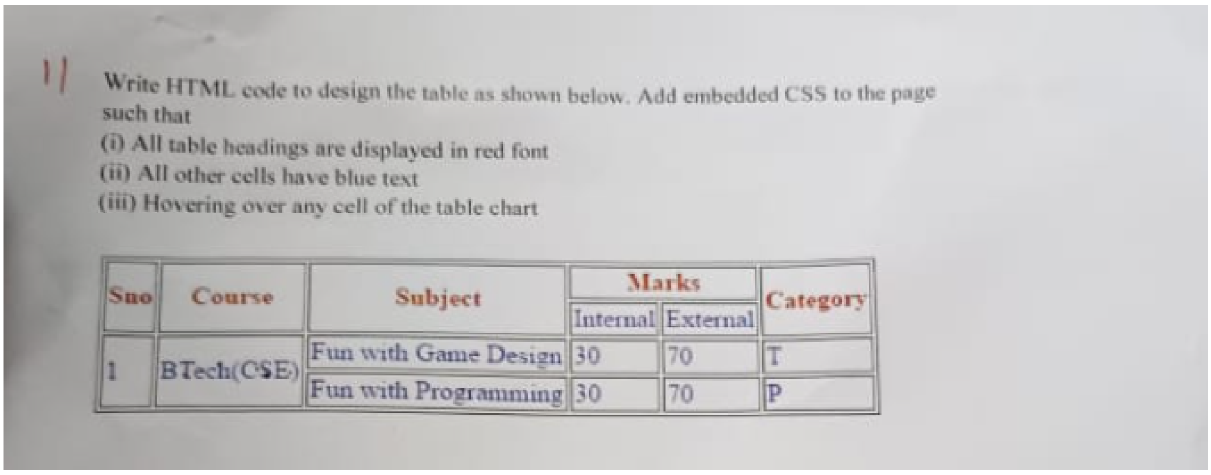
"Write HTML and CSS code to create an invoice table with the following structure:
- Header row with 'Invoice #123456789' and '14 January 2025'
- Pay to and Customer sections with addresses
- Item table with columns: Name/Description, Qty., @, Cost
- Items: Paper clips (1000 @ 0.01), Staples (100 @ 1.00)
- Footer with Subtotal (110.00), Tax (8%, 8.80), and Grand Total ($118.80)
- Apply document-level CSS to make heading cells bold with grey background"


17) Write a JavaScript function to take a string as input parameter and reverse the case of every alphabet in the string. The function should return the modified string.
Ex. changeCase ("HeLlo") must return "hElLO"

```javascript
function changeCase(inputString) {
    let result = '';
    for (let i = 0; i < inputString.length; i++) {
        const char = inputString[i];
        if (char === char.toUpperCase() && char !== char.toLowerCase()) {
            result += char.toLowerCase();
        } else if (char === char.toLowerCase() && char !== char.toUpperCase()) {
            result += char.toUpperCase();
        } else {
            result += char;
        }
    }
    return result;
}

// Example usage:
console.log(changeCase("Hello"));  // Output: "hELLO"
```


8) A HTML page contains the following:
- A table exists with id "table1"
- A button "Add" with id "btn1"
- A div with id "display"
Add JavaScript to the webpage for the following specifications
• On clicking "Add" a new row with 2 cells are added to the table
• Populate each cell with a random number (between 1 and 200)
• If the cell contains an even number, mouse over should turn the cell green
• If the cell contains an odd number, mouse over should turn the cell red.
• If you click on a cell, the div will be populated with the cell content

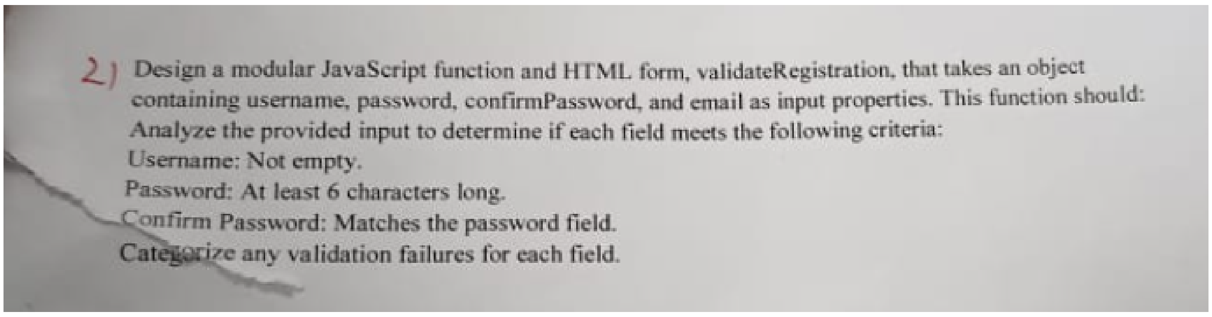
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Table</title>
    <style>
        table {
            border-collapse: collapse;
            width: 50%;
            margin: 20px 0;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: center;
            cursor: pointer;
        }
        th {
            background-color: #f2f2f2;
        }
        #display {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ccc;
            min-height: 20px;
            width: 50%;
        }
    </style>
</head>
<body>
    <table id="table1">
        <tr>
            <th>Column 1</th>
            <th>Column 2</th>
        </tr>
    </table>
    <button id="btn1">Add Row</button>
    <div id="display"></div>

    <script>
        document.getElementById('btn1').addEventListener('click', function() {
            const table = document.getElementById('table1');
            const newRow = table.insertRow();

            // Create two cells with random numbers
            for (let i = 0; i < 2; i++) {
                const cell = newRow.insertCell();
                const randomNum = Math.floor(Math.random() * 200) + 1;
                cell.textContent = randomNum;

                // Add mouseover event for color change
                cell.addEventListener('mouseover', function() {
                    if (randomNum % 2 === 0) {
                        this.style.backgroundColor = 'lightgreen';
                    } else {
                        this.style.backgroundColor = 'lightcoral';
                    }
                });

                // Add mouseout event to clear color
                cell.addEventListener('mouseout', function() {
                    this.style.backgroundColor = '';
                });

                // Add click event to display content
                cell.addEventListener('click', function() {
                    document.getElementById('display').textContent = this.textContent;
                });
            }
        });
    </script>
</body>
</html>
```


1) Write HTML code to design the table as shown below. Add embedded CSS to the page such that
(i) All table headings are displayed in red font
(ii) All other cells have blue text
(iii) Hovering over any cell of the table chart

Sno	Course	Subject	Marks		Category
			Internal	External	
1	BTech(CSE)	Fun with Game Design	30	70	T
		Fun with Programming	30	70	P

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Academic Table</title>
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
      font-family: Arial, sans-serif;
      margin: 20px 0;
    }

    th {
      color: red;
      font-weight: bold;
      border: 1px solid #ddd;
      padding: 8px;
      text-align: left;
    }

    td {
      color: blue;
      border: 1px solid #ddd;
      padding: 8px;
    }

    tr:hover td {
      background-color: #f5f5f5;
    }

    /* Special styling for merged cells */
    .merged-row {
      border-top: none;
```

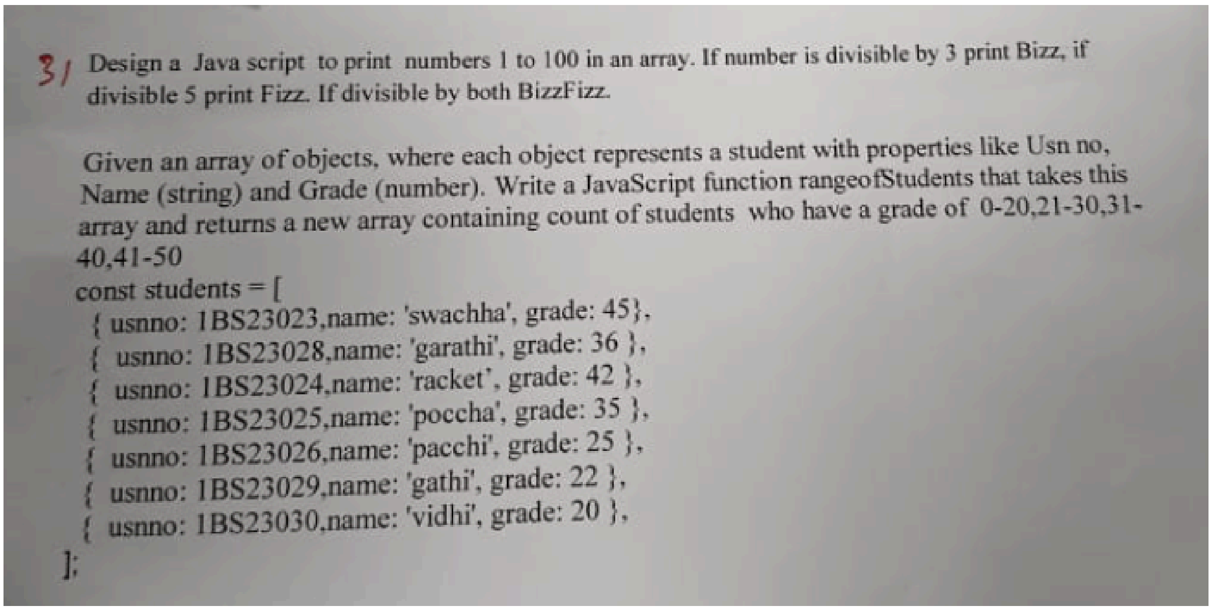
```
        }
    </style>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th>Sno</th>
                <th>Course</th>
                <th>Subject</th>
                <th colspan="2">Marks</th>
                <th>Category</th>
            </tr>
            <tr>
                <td></td>
                <td></td>
                <td></td>
                <th>Internal</th>
                <th>External</th>
                <td></td>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>BTech(CSE)</td>
                <td>Fun with Game Design</td>
                <td>30</td>
                <td>70</td>
                <td rowspan="2"></td>
            </tr>
            <tr class="merged-row">
                <td></td>
                <td></td>
                <td>Fun with Programming</td>
                <td>30</td>
                <td></td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```



2) Design a modular JavaScript function and HTML form, validateRegistration, that takes an object containing username, password, confirmPassword, and email as input properties. This function should: Analyze the provided input to determine if each field meets the following criteria:
Username: Not empty.
Password: At least 6 characters long.
Confirm Password: Matches the password field.
Categorize any validation failures for each field.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Registration Form Validation</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         max-width: 500px;
         margin: 0 auto;
         padding: 20px;
      }
      .form-group {
         margin-bottom: 15px;
      }
      label {
         display: block;
         margin-bottom: 5px;
         font-weight: bold;
      }
      input {
         width: 100%;
         padding: 8px;
         box-sizing: border-box;
      }
      button {
         padding: 10px 15px;
         background-color: #4CAF50;
         color: white;
         border: none;
         cursor: pointer;
      }
      .error {
         color: red;
         font-size: 0.9em;
         margin-top: 5px;
      }
```

```
    </style>
</head>
<body>
    <h1>Registration Form</h1>
    <form id="registrationForm">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username">
            <div id="usernameError" class="error"></div>
        </div>

        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password">
            <div id="passwordError" class="error"></div>
        </div>

        <div class="form-group">
            <label for="confirmPassword">Confirm Password:</label>
            <input type="password" id="confirmPassword" name="confirmPassword">
            <div id="confirmPasswordError" class="error"></div>
        </div>

        <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email">
            <div id="emailError" class="error"></div>
        </div>

        <button type="submit">Register</button>
    </form>

    <script>
        // Modular validation function
        function validateRegistration(userData) {
            const errors = {};

            // Username validation
            if (!userData.username.trim()) {
                errors.username = "Username is required";
            }

            // Password validation
            if (userData.password.length < 6) {
                errors.password = "Password must be at least 6 characters";
            }

            // Confirm Password validation
            if (userData.password !== userData.confirmPassword) {
                errors.confirmPassword = "Passwords do not match";
            }

            // Email validation
            if (!userData.email.includes('@') || !userData.email.includes('.')) {
                errors.email = "Please enter a valid email address";
            }

            return {
                isValid: Object.keys(errors).length === 0,
                errors
            };
        }

        // Form submission handler
        document.getElementById('registrationForm').addEventListener('submit', function(e) {
            e.preventDefault();

            // Clear previous errors
            document.querySelectorAll('.error').forEach(el => {
                el.textContent = '';
            });

            // Get form data
            const formData = {
                username: document.getElementById('username').value,
                password: document.getElementById('password').value,
                confirmPassword: document.getElementById('confirmPassword').value,
                email: document.getElementById('email').value
            };

            // Validate
            const validationResult = validateRegistration(formData);

            if (validationResult.isValid) {
                alert('Registration successful!');
                // Here you would typically submit the form to a server
            } else {
                // Display errors
                for (const [field, message] of Object.entries(validationResult.errors)) {
                    document.getElementById(`${field}Error`).textContent = message;
                }
            }
        });
    </script>
</body>
</html>
```


3/ Design a Java script to print numbers 1 to 100 in an array. If number is divisible by 3 print Bizz, if divisible 5 print Fizz. If divisible by both BizzFizz.

Given an array of objects, where each object represents a student with properties like Usn no, Name (string) and Grade (number). Write a JavaScript function rangeofStudents that takes this array and returns a new array containing count of students who have a grade of 0-20,21-30,31-40,41-50

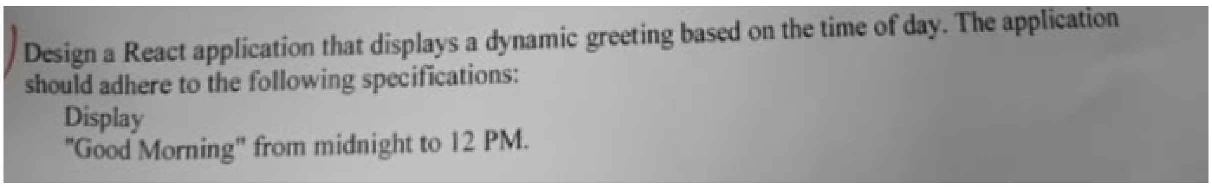
```
const students = [
 { usnno: 1BS23023,name: 'swachha', grade: 45},
 { usnno: 1BS23028,name: 'garathi', grade: 36 },
 { usnno: 1BS23024,name: 'racket', grade: 42 },
 { usnno: 1BS23025,name: 'poccha', grade: 35 },
 { usnno: 1BS23026,name: 'pacchi', grade: 25 },
 { usnno: 1BS23029,name: 'gathi', grade: 22 },
 { usnno: 1BS23030,name: 'vidhi', grade: 20 },
];
```

```
function bizzFizz() {
  const result = [];
  for (let i = 1; i <= 100; i++) {
    if (i % 3 === 0 && i % 5 === 0) {
      result.push("BizzFizz");
    } else if (i % 3 === 0) {
      result.push("Bizz");
    } else if (i % 5 === 0) {
      result.push("Fizz");
    } else {
      result.push(i);
    }
  }
  return result;
}

// Usage:
const bizzFizzArray = bizzFizz();
console.log(bizzFizzArray);
```

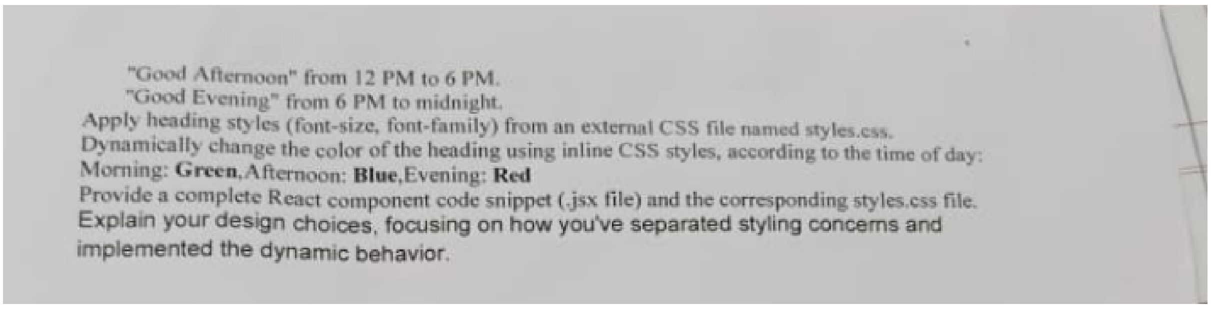
2nd part

```
const students = [
  { usnno: 'IBS23023', name: 'swachha', grade: 45 },
  { usnno: 'IBS23028', name: 'garathi', grade: 36 },
  { usnno: 'IBS23024', name: 'racket', grade: 42 },
  { usnno: 'IBS23025', name: 'poccha', grade: 35 },
  { usnno: 'IBS23026', name: 'pacchi', grade: 25 },
  { usnno: 'IBS23029', name: 'gathi', grade: 22 },
  { usnno: 'IBS23030', name: 'vidhi', grade: 20 }
];
```

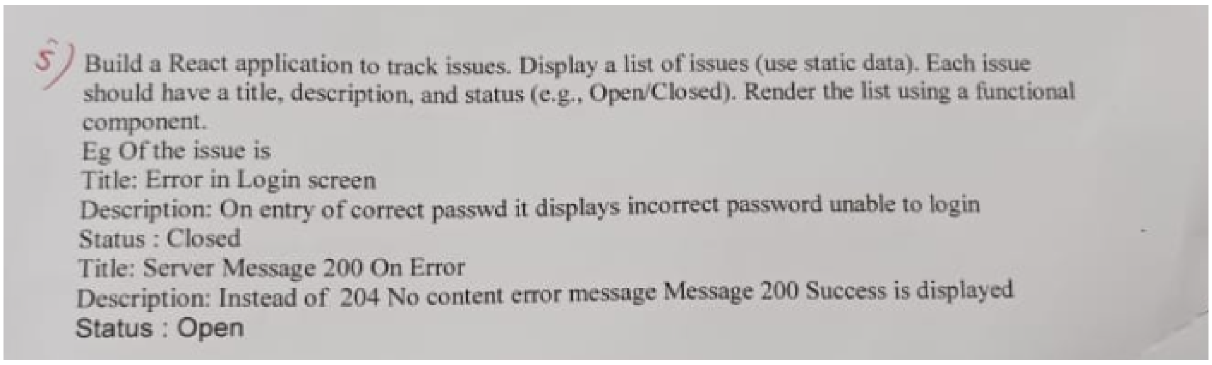
```
function rangeOfStudents(studentArray) {
    const count = {
        '0-20': 0,
        '21-30': 0,
        '31-40': 0,
        '41-50': 0
    };

    studentArray.forEach(student => {
        const grade = student.grade;
        if (grade >= 0 && grade <= 20) {
            count['0-20']++;
        } else if (grade >= 21 && grade <= 30) {
            count['21-30']++;
        } else if (grade >= 31 && grade <= 40) {
            count['31-40']++;
        } else if (grade >= 41 && grade <= 50) {
            count['41-50']++;
        }
    });

    return count;
}

// Usage:
const gradeRanges = rangeOfStudents(students);
console.log(gradeRanges);
```


Design a React application that displays a dynamic greeting based on the time of day. The application should adhere to the following specifications:

Display
"Good Morning" from midnight to 12 PM.

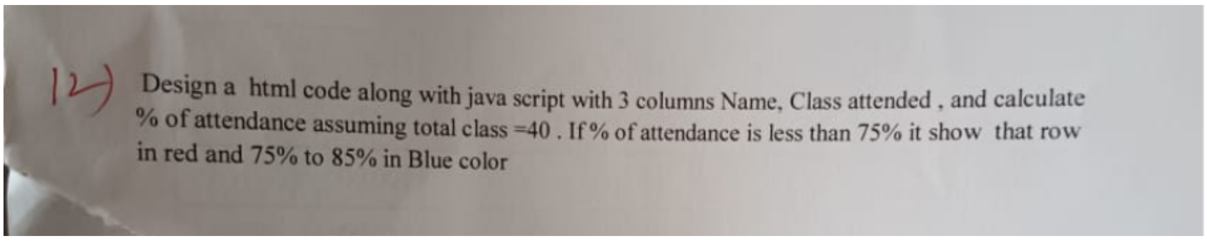

"Good Afternoon" from 12 PM to 6 PM.
"Good Evening" from 6 PM to midnight.
Apply heading styles (font-size, font-family) from an external CSS file named styles.css.
Dynamically change the color of the heading using inline CSS styles, according to the time of day:
Morning: **Green**,Afternoon: **Blue**,Evening: **Red**
Provide a complete React component code snippet (.jsx file) and the corresponding styles.css file.
Explain your design choices, focusing on how you've separated styling concerns and implemented the dynamic behavior.

```
import React, { useState, useEffect } from 'react';

function TimeBasedGreeting() {
  const [greeting, setGreeting] = useState('');
  useEffect(() => {
    const updateGreeting = () => {
      const now = new Date();
      const hours = now.getHours();

      if (hours >= 0 && hours < 12) {
        setGreeting('Good Morning');
      } else if (hours >= 12 && hours < 17) {
        setGreeting('Good Afternoon');
      } else if (hours >= 17 && hours < 21) {
        setGreeting('Good Evening');
      } else {
        setGreeting('Good Night');
      }
    };

    // Update greeting immediately
    updateGreeting();

    // Update greeting every minute to handle day changes
    const interval = setInterval(updateGreeting, 60000);

    return () => clearInterval(interval);
  }, []);

  return (
    <div className="greeting-app">
      <h1>{greeting}</h1>
      <p>Current time: {new Date().toLocaleTimeString()}</p>
    </div>
  );
}

export default TimeBasedGreeting;
```


5) Build a React application to track issues. Display a list of issues (use static data). Each issue should have a title, description, and status (e.g., Open/Closed). Render the list using a functional component.
Eg Of the issue is
Title: Error in Login screen
Description: On entry of correct passwd it displays incorrect password unable to login
Status : Closed
Title: Server Message 200 On Error
Description: Instead of 204 No content error message Message 200 Success is displayed
Status : Open

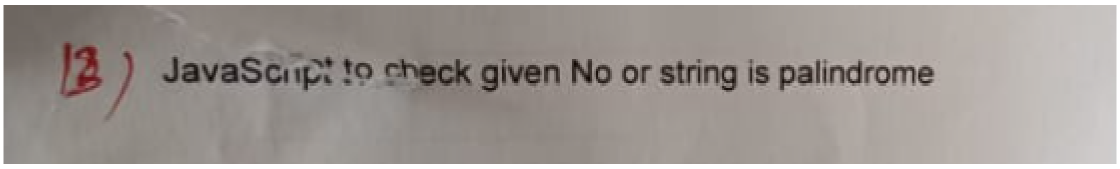
```
import React from 'react';
import './App.css';
```

```
function IssueTracker() {
  // Static issue data
  const issues = [
    {
      id: 1,
      title: 'Error in Login screen',
      description: 'On entry of correct password it displays incorrect password unable to login',
      status: 'Closed'
    },
    {
      id: 2,
      title: 'Server Message 200 On Error',
      description: 'Instead of 204 No content error message Message 200 Success is
displayed',
      status: 'Open'
    },
    {
      id: 3,
      title: 'Mobile Responsiveness Issue',
      description: 'Layout breaks on mobile devices below 400px width',
      status: 'Open'
    }
  ];

  return (
    <div className="issue-tracker">
      <h1>Issue Tracker</h1>
      <div className="issues-container">
        {issues.map(issue => (
          <div key={issue.id} className={`issue-card ${issue.status.toLowerCase()}`}>
            <h3>{issue.title}</h3>
            <p className="description">{issue.description}</p>
            <div className="status">
              Status: <span className={`status-badge
${issue.status.toLowerCase()}`}>{issue.status}</span>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
}

export default IssueTracker;
```


12) Design a html code along with java script with 3 columns Name, Class attended , and calculate % of attendance assuming total class =40 . If % of attendance is less than 75% it show that row in red and 75% to 85% in Blue color

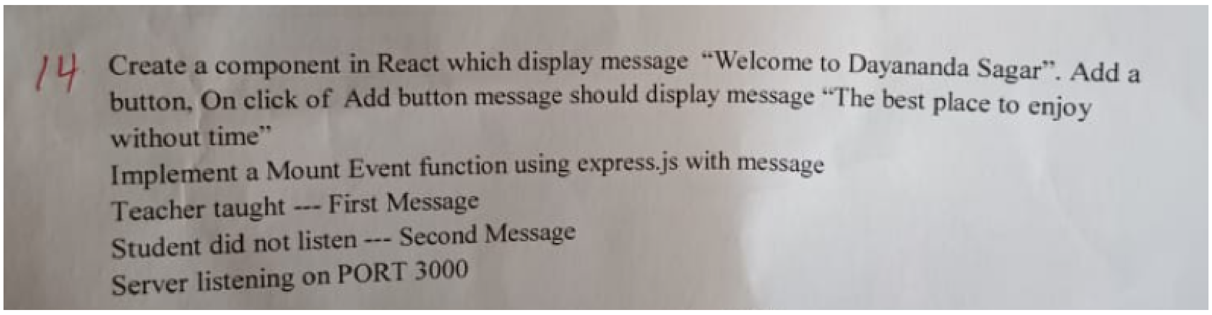
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Attendance Tracker</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    table {
      width: 80%;
      border-collapse: collapse;
      margin: 20px auto;
    }
    th, td {
      border: 1px solid #ddd;
      padding: 8px;
      text-align: center;
    }
    th {
      background-color: #f2f2f2;
    }
    .low-attendance {
      background-color: #ffcccc; /* Red for <75% */
    }
    .medium-attendance {
      background-color: #cce5ff; /* Blue for 75-85% */
    }
    .high-attendance {
      background-color: #ccffcc; /* Green for >85% */
    }
    h1 {
      text-align: center;
      color: #333;
    }
  </style>
</head>
<body>
  <h1>Attendance Report</h1>
```

```
    <table id="attendanceTable">
        <thead>
            <tr>
                <th>Name</th>
                <th>Classes Attended</th>
                <th>Attendance %</th>
            </tr>
        </thead>
        <tbody>
            <!-- Data will be inserted by JavaScript -->
        </tbody>
    </table>

    <script>
        // Sample data - can be replaced with actual data
        const students = [
            { name: "John Doe", attended: 25 },
            { name: "Jane Smith", attended: 32 },
            { name: "Mike Johnson", attended: 30 },
            { name: "Sarah Williams", attended: 28 },
            { name: "David Brown", attended: 35 }
        ];

        const totalClasses = 40;
        const tableBody = document.querySelector('#attendanceTable tbody');

        students.forEach(student => {
            // Calculate attendance percentage
            const percentage = (student.attended / totalClasses) * 100;
            const roundedPercentage = Math.round(percentage * 10) / 10; // Round to 1 decimal
place

            // Create new row
            const row = document.createElement('tr');

            // Determine row class based on attendance percentage
            if (percentage < 75) {
                row.classList.add('low-attendance');
            } else if (percentage >= 75 && percentage <= 85) {
                row.classList.add('medium-attendance');
            } else {
                row.classList.add('high-attendance');
            }

            // Add cells to the row
            row.innerHTML = `
                <td>${student.name}</td>
                <td>${student.attended}</td>
            <td>${roundedPercentage}%</td>
          `;

          // Add row to table
          tableBody.appendChild(row);
        });
    </script>
</body>
</html>
```



13) JavaScript to check given No or string is palindrome

```
function isPalindrome(input) {
    // Convert input to string to handle both numbers and strings
    const str = String(input).toLowerCase().replace(/[^a-z0-9]/g, '');

    // Compare the string with its reverse
    return str === str.split('').reverse().join('');
}
```

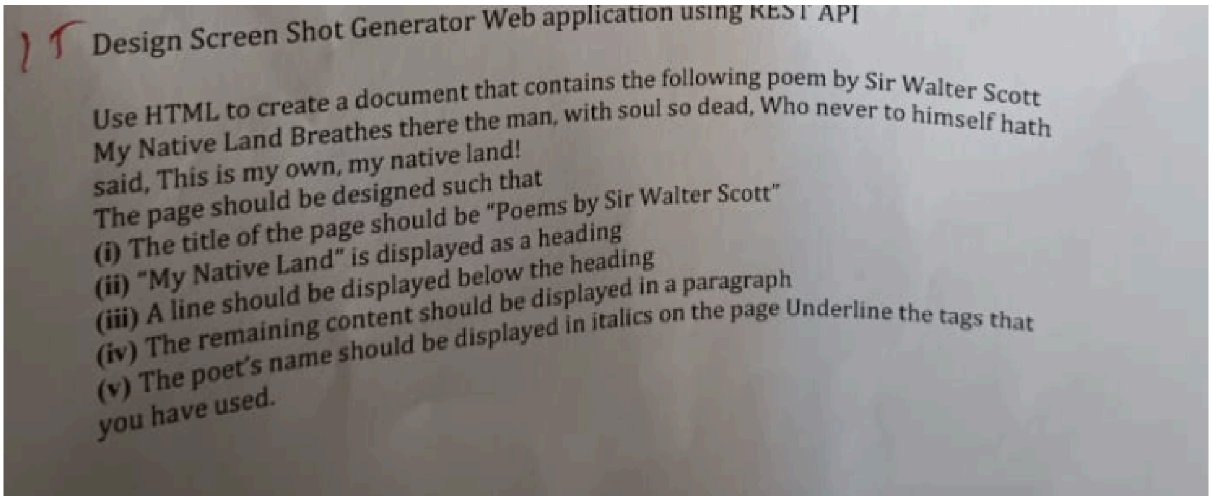


14 Create a component in React which display message "Welcome to Dayananda Sagar". Add a button. On click of Add button message should display message "The best place to enjoy without time"
Implement a Mount Event function using express.js with message
Teacher taught --- First Message
Student did not listen --- Second Message
Server listening on PORT 3000

```
import React, { useState } from 'react';

function WelcomeMessage() {
  const [message, setMessage] = useState("Welcome to Dayananda Sagar");

  const handleClick = () => {
    setMessage("The best place to enjoy without time");
  };

  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>{message}</h1>
      <button
        onClick={handleClick}
        style={{
          padding: '10px 20px',
          fontSize: '16px',
          backgroundColor: '#4CAF50',
          color: 'white',
          border: 'none',
          borderRadius: '5px',
          cursor: 'pointer'
        }}
      >
        Change Message
      </button>
    </div>
  );
}

export default WelcomeMessage;
```

. Express.js Server (Mount Event Function)

```
const express = require('express');
const app = express();

// Mount event middleware
app.use((req, res, next) => {
  console.log("Teacher taught --- First Message");
  next();
  console.log("Student did not listen --- Second Message");
});

// Basic route
app.get('/', (req, res) => {
  res.send('Server is running');
});

// Start server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server listening on PORT ${PORT}`);
});
```



) T Design Screen Shot Generator Web application using REST API

Use HTML to create a document that contains the following poem by Sir Walter Scott
My Native Land Breathes there the man, with soul so dead, Who never to himself hath said, This is my own, my native land!
The page should be designed such that
(i) The title of the page should be "Poems by Sir Walter Scott"
(ii) "My Native Land" is displayed as a heading
(iii) A line should be displayed below the heading
(iv) The remaining content should be displayed in a paragraph
(v) The poet's name should be displayed in italics on the page Underline the tags that you have used.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Poems by Sir Walter Scott</title>
    <style>
        body {
            font-family: 'Georgia', serif;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
            line-height: 1.8;
            background-color: #f9f9f9;
        }
        h1 {
            text-align: center;
            color: #333;
            margin-bottom: 30px;
            text-decoration: underline;
        }
        h2 {
            color: #444;
            text-align: center;
            text-decoration: underline;
            margin-bottom: 10px;
        }
        hr {
            width: 200px;
            margin: 0 auto 20px;
            border: 0;
            height: 1px;
            background: linear-gradient(to right, transparent, #333, transparent);
        }
        p {
            font-size: 1.1em;
            text-align: center;
            margin: 20px 0;
        }
        .poet {
            font-style: italic;
            text-align: right;
            margin-top: 40px;
            text-decoration: underline;
        }
        .screenshot-btn {
            display: block;
            width: 200px;
            margin: 30px auto;
            padding: 10px;
            background-color: #4a6fa5;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1><u>Poems by Sir Walter Scott</u></h1>

    <div id="poem-content">
        <h2><u>My Native Land</u></h2>
        <hr>
        <p>
            <u>Breathes</u> there the man, with soul so dead,<br>
            Who never to himself hath said,<br>
            This is my own, my native land!
        </p>

        <div class="poet"><u>- Sir Walter Scott</u></div>
    </div>

    <button class="screenshot-btn" id="capture-btn">Generate Screenshot</button>

    <script>
        document.getElementById('capture-btn').addEventListener('click', async () => {
            try {
                const response = await fetch('https://api.screenshotapi.io/capture', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': 'Bearer YOUR_API_KEY' // Replace with actual API key
                    },
                    body: JSON.stringify({
                        url: window.location.href,
                        viewport: {
                            width: 800,
                            height: 600
                        },
                        full_page: true
                    })
                });

                const data = await response.json();
                if (data.screenshot_url) {
                    window.open(data.screenshot_url, '_blank');
                } else {
                    alert('Screenshot generation failed. Please try again.');
                }
            } catch (error) {
                console.error('Error:', error);
                alert('Failed to generate screenshot. Check console for details.');
            }
        });
    </script>
</body>
</html>
```