|  |
| --- |
| Serial No: |
| **Sessional Exam-II** |
| **Total Time: 1 Hour** |
| **Total Marks: 60** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Signature of Invigilator |

|  |
| --- |
| **CS-4032: Web Programming(BSCS-A,B)** |
| Monday, 5th April, 2024 |
| **Course Instructor** |
| Mr. Aqib Rehman |

|  |
| --- |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Student Name Roll No. Course Section Student Signature |

## DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

**Instructions:**

1. Verify at the start of exam that you have a total of **three (3)** questions printed on **thirteen (13)** pages including this title page.
2. Attempt all questions on the question-book and in the given order.
3. This exam is **closed book**. Mobiles, Internet and note-sharing is not allowed. Please see that the area in your threshold is free of any material classified *as useful in the paper*, i.e. mobile/internet or else there may be a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Fit in all your answers in the provided space. You may use extra space on last page if required. If you do so, clearly mark question/ part number on that page to avoid confusion.
6. Use only your own stationary.
7. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part done using soft pencil cannot be claimed for checking/rechecking.

"I can accept failure. Everyone fails at something. But I can't accept not trying." ― Michael Jordan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Q-1** | **Q-2** | **Q-3** | **Total** |
| **Marks Obtained** |  |  |  |  |
| **Total**  **Marks** | **26** | **16** | **18** | **60** |

**Question 1 [4\*4+5+5=26 Marks]**

1. **Identify the difference between Real DOM and Virtual DOM. State the Virtual DOM working. [4 Marks]**

The main difference between the Real DOM (Document Object Model) and the Virtual DOM lies in their implementations and how they interact with the browser.

**Real DOM:**

The Real DOM represents the actual HTML structure of a web page, created by the browser.

It is a tree-like structure of objects that corresponds directly to the elements in the HTML document.

Whenever there is a change in the state of the application, the Real DOM updates the affected parts of the web page directly.

Manipulating the Real DOM can be slow and inefficient, especially for large and complex web applications, as every change triggers reflows and repaints, impacting performance.

**Virtual DOM:**

The Virtual DOM is an abstraction of the Real DOM, maintained in memory by JavaScript libraries like React.

It is a lightweight copy of the Real DOM, representing the current state of the web page.

When there is a change in the state of the application, the Virtual DOM updates itself first, without touching the Real DOM.

After updating the Virtual DOM, React compares it with the previous version (reconciliation) to determine the minimum set of changes needed to update the Real DOM.

Only the differences between the old and new versions of the Virtual DOM are applied to the Real DOM, reducing the number of DOM manipulations and improving performance.

1. **List the events in React and how do you create an event in React? [4 Marks]**

list of common events in React:

**Mouse Events:**

onClick: Triggered when a user clicks an element.

onDoubleClick: Triggered when a user double-clicks an element.

onMouseDown: Triggered when a mouse button is pressed over an element.

onMouseUp: Triggered when a mouse button is released over an element.

onMouseMove: Triggered when the mouse pointer moves within an element.

onMouseEnter: Triggered when the mouse pointer enters an element.

onMouseLeave: Triggered when the mouse pointer leaves an element.

**Keyboard Events:**

onKeyDown: Triggered when a key is pressed down.

onKeyPress: Triggered when a key is pressed and released.

onKeyUp: Triggered when a key is released.

**Form Events:**

onChange: Triggered when the value of an input, select, or textarea element changes.

onSubmit: Triggered when a form is submitted.

**Etc.**

import React from 'react';

function ButtonWithEvent() {

// Define a function to handle the onClick event

function handleClick() {

alert('Button clicked!');

}

// Render a button element with onClick event handler

return (

<button onClick={handleClick}>

Click me

</button>

);

}

export default ButtonWithEvent;

1. **What is keys props? And Is it necessary to always use keys props in React. Provide your point of view. [4 Marks]**

In React, the key prop is a special attribute used to uniquely identify elements in a list or iterable. It helps React identify which items have changed, are added, or are removed from a list during rendering. The key prop is required when rendering lists of elements using the map() function or when dynamically generating elements in a component.

1. **Discuss the types of creating a component in React and provide an example code. [4 Marks]**

**Note: No marks without example code.**

**Functional Components:**

**import React from 'react';**

**const FunctionalComponent = (props) => {**

**return (**

**<div>**

**<h1>Hello, {props.name}!</h1>**

**<p>{props.message}</p>**

**</div>**

**);**

**}**

**export default FunctionalComponent;**

**Class Components:**

import React, { Component } from 'react';

class ClassComponent extends Component {

constructor(props) {

super(props);

this.state = {

count: 0

};

}

render() {

return (

<div>

<h1>Count: {this.state.count}</h1>

<button onClick={() => this.setState({ count: this.state.count + 1 })}>

Increment

</button>

</div>

);

}

}

export default ClassComponent;

1. **What will the code below output to the console and why? [5 Marks]**

**var** arr1 = "web".**split**('');

**var** arr2 = arr1.**reverse**();

**var** arr3 = "Programming".**split**('');

arr2.**push**(arr3);

console.**log**("array 2: "+arr2);

arr2.**unshift**("web");

console.**log**("array 2 updated: "+arr2);

console.**log**("array 1: length=" + arr1.length + " val=" + arr1.**slice**(2));

console.**log**("array 2: length=" + arr2.length + " get=" + arr2.**slice**(-1));

Output:

array 2: b,e,w,P,r,o,g,r,a,m,m,i,n,g

array 2 updated: web,b,e,w,P,r,o,g,r,a,m,m,i,n,g

array 1: length=5 val=e,w,P,r,o,g,r,a,m,m,i,n,g

array 2: length=5 get=P,r,o,g,r,a,m,m,i,n,g

1. **What will the code below output to the console and why? [5 Marks]**

**var x = 21;**

**var foo = function () {**

**console.log(x);**

**var x = 20;**

**};**

**foo ();**

**const mult = (a, b, c) => {**

**a = a !== undefined ? a : 0;**

**b = b !== undefined ? b : 1;**

**return a \* b \* c;**

**}**

**console.log(mult(2, 3));**

Output:

undefined

NaN

**Question 2 [8+8=16 Marks]**

1. **Write the complete code for a JavaScript app that takes a number input from user to display a particular string e.g. your name that many times. For example If user enter 10 the code should display his/her name 10 times in an unordered html list. [8 Marks]**

<body>

<h2>Enter the number of times to display your name:</h2>

<input type="number" id="inputNumber">

<button onclick="displayName()">Display</button>

<ul id="nameList"></ul>

<script>

function displayName() {

const inputNumber = document.getElementById('inputNumber').value;

const nameList = document.getElementById('nameList');

nameList.innerHTML = '';

for (let i = 0; i < inputNumber; i++) {

const listItem = document.createElement('li');

listItem.textContent = 'Your Name';

nameList.appendChild(listItem);

}

}

</script>

</body>

</html>

1. **Consider the code of Question 2 part i and convert it in React app without using create-react-app package. Design a class base Component with name “App” and define the inputNo and isEven variable in the state. By clicking on “check\_it” button, call the user defined function with name “check\_function”. In the function check the entered no. and render the component in such an order that it shows the entered no and its result of even or not on the page to the end user (Hint: Change the state mechanism). [8 Marks]**
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">
6. <meta http-equiv="X-UA-Compatible" content="ie=edge">
7. <title>React Basics</title>
8. <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
9. <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
10. <script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
11. </head>
12. <body>
13. <div id="app"></div>
14. <script type="text/babel">
15. class App extends React.Component {
16. state = {
17. inputNo:0,
18. isEven:"even"
19. }
20. handleClick2 = (e) => {
21. console.log(e.target);
22. const v = document.getElementById('req').value;
23. if(v%2==0){
24. this.setState({
25. isEven: "even"
26. });
27. }
28. else{
29. this.setState({
30. isEven: "odd"
31. });
32. }
33. this.setState({
34. inputNo: v
35. });
36. }
38. render(){
39. return(
40. <div className="app-content">
42. <h1>input No. {this.state.inputNo}</h1>
43. <h1>Result {this.state.isEven}</h1>
44. <form>
45. <input type="number" id="req" />
46. <button onClick={this.handleClick2}>Check it</button>
48. </form>
49. </div>
50. )
51. }
52. }
53. ReactDOM.render(<App />, document.getElementById('app'));
54. </script>
55. </body>
56. </html>

**Question 3 [18 Marks]**

**Develop a quiz game in React library with random mathematical expressions using addition, subtraction multiplication, division and modulus operators. Both the operations and operands must be shown randomly in the quiz questions. The game must have a scoring state which will be initially at Score: 0. The score gets incremented on every correct answer. Also penalize score by negative 1 for wrong answer. The game will reach winning state at the total score of 10.**

**Write a command, which is used to create a new react app using Node package. [1 Mark]**

npx create-react-app quizgame

**After creation of a react app what will be the directory structure. [2 Marks]**

src,public

**Write a command to run the react app in browser local development server. [1 Mark]**

npm start

**Write the code of index.html file. [2 Marks]**

<body>

<div id="root"></div>

</body>

</html>

**Write the code of index.js file. [2 Marks]**

const root = ReactDOM.createRoot(document.getElementById('root')).render(<App />);

**Write the code of required components here. [10 Marks]**

import React, { useState } from 'react';

const QuizGame = () => {

const [score, setScore] = useState(0);

const [expression, setExpression] = useState('');

const [answer, setAnswer] = useState('');

const [result, setResult] = useState('');

const [winningState, setWinningState] = useState(false);

// Function to generate random mathematical expressions

const generateExpression = () => {

const operators = ['+', '-', '\*', '/', '%'];

const randomOperator = operators[Math.floor(Math.random() \* operators.length)];

const randomNumber1 = Math.floor(Math.random() \* 10) + 1;

const randomNumber2 = Math.floor(Math.random() \* 10) + 1;

setExpression(`${randomNumber1} ${randomOperator} ${randomNumber2}`);

};

// Function to handle user input and check answer

const handleAnswerSubmit = () => {

const correctAnswer = eval(expression);

if (parseInt(answer) === correctAnswer) {

setScore(score + 1);

setResult('Correct!');

} else {

setScore(score - 1);

setResult('Incorrect!');

}

if (score + 1 === 10) {

setWinningState(true);

}

setAnswer('');

generateExpression();

};

// Function to reset the game

const resetGame = () => {

setScore(0);

setWinningState(false);

generateExpression();

};

// Initialize the first expression

if (expression === '') {

generateExpression();

}

return (

<div>

<h1>Quiz Game</h1>

<p>Score: {score}</p>

<p>{expression}</p>

<input

type="number"

value={answer}

onChange={(e) => setAnswer(e.target.value)}

/>

<button onClick={handleAnswerSubmit}>Submit</button>

<p>{result}</p>

{winningState && <p>Congratulations! You've reached the winning score!</p>}

<button onClick={resetGame}>Reset</button>

</div>

);

};

export default QuizGame;

**Best of Luck ☺**