|  |
| --- |
| **KONGU ENGINEERING COLLEGE, PERUNDURAI - 638 060** |
| **SEMESTER  ODD|CONTINUOUS ASSESSMENT TEST – II** |
| (Regulations **2020**) **Answer Key** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Month and Year | : | August2023 | Roll Number | : |  |
| Programme | : | B.Tech. | Date | : | 09.10.2023 |
| Branch | : | IT | Time | : | 02.30pm - 04.00pm |
| Semester | : | V | Duration | : | 1½ Hours |
| Course Code | : | 20ITE03 | Max. Marks | : | 50 |
| Course Name | : | User Interface Design |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PART - A   (10 × 2 = 20 Marks)** | | | | | |
| **ANSWER ALL THE QUESTIONS** | | | | | |
|  | Consider the following reusable component and Provide a TaskList component which reuses the above Task Component  // Task.js  import React from 'react';  const Task = ({ title, description }) => {    return (  <div className="task">     <h3>{title}</h3>     <p>{description}</p>  </div>    );  };  export default Task;  Answer:  import React from 'react';  import Task from './Task'; // Assuming Task.js is in the same directory or correct path  const TaskList = ({ tasks }) => {    return (      <div className="task-list">        {tasks.map((task, index) => (          <Task            key={index} // You should use a unique identifier as the key if you have one            title={task.title}            description={task.description}          />        ))}      </div>    );  };  export default TaskList; | | | CO2 | K3 |
|  | Give an Example for styled-components.  npm install styled-components  import React from 'react';  import styled from 'styled-components';  // Create a styled component  const StyledButton = styled.button`    background-color: #007bff;    color: #fff;    border: none;    padding: 10px 20px;    `;  // Create a functional React component  function MyComponent() {    return (      <div>             <StyledButton>Click Me</StyledButton>      </div>    );  }  export default MyComponent; | | | CO2 | K1 |
|  | Tell the tag which is used to display the web pages in mobile devices.  <meta name="viewport" content="width=device-width, initial-scale=1"> | | | CO2 | K1 |
|  | Mention the purpose of using PropTypes in React, and what types of validation can you perform with them?  PropTypes in React are used to add a level of type checking to your components, ensuring that the data being passed between components is of the correct type.  PropTypes allow you to specify the expected types of the props that your components should receive. This helps catch runtime errors and bugs early during development, making your code more robust.  Primitive Types:      PropTypes.string: Validates that the prop is a string.      PropTypes.number: Validates that the prop is a number.      PropTypes.boolean: Validates that the prop is a boolean.  import React from 'react';  import PropTypes from 'prop-types';  function MyComponent(props) {    return (      <div>        <h1>{props.title}</h1>        <p>{props.description}</p>      </div>    );  }  MyComponent.propTypes = {    title: PropTypes.string.isRequired,    description: PropTypes.string,  };  export default MyComponent; | | | CO3 | K1 |
|  | Compose a simple profile components using props(Minimum 3 Components must be nested all with Props)  import React from 'react';  function ProfileImage(props) {    return <img src={props.imageSrc} alt={props.name} />;  }  export default ProfileImage;  import React from 'react';  function ProfileInfo(props) {    return (      <div>        <h2>{props.name}</h2>        <p>{props.bio}</p>      </div>    );  }  export default ProfileInfo;  import React from 'react';  import ProfileImage from './ProfileImage';  import ProfileInfo from './ProfileInfo';  function Profile(props) {    return (      <div>        <ProfileImage imageSrc={props.imageSrc} name={props.name} />        <ProfileInfo name={props.name} bio={props.bio} />      </div>    );  }  export default Profile;  import React from 'react';  import Profile from './Profile'; // Assuming the components are in separate files  function App() {    const user = {      name: 'John Doe',      bio: 'Front-end Developer',      imageSrc: 'profile.jpg',    };    return (    <div>        <h1>User Profile</h1>        <Profile {...user} />      </div>  );}  export default App; | | | CO3 | K2 |
|  | Compare this.state with usestate.   |  |  | | --- | --- | | Class Components (this.state) | Functional Components with useState | | this.state is used in class components to manage local component state.  It's an object that holds the state data for the component.  State changes are made using this.setState(), and React handles re-rendering the component when the state changes. | useState is a React Hook used in functional components to manage state.  It's a function that returns an array with two elements: the current state and a function to update the state.  State changes are made by calling the state updater function returned by useState, and React re-renders the component when the state changes. | | class MyComponent extends React.Component {    constructor() {      super();      this.state = {        count: 0,      };    }    incrementCount = () => {      this.setState({ count: this.state.count + 1 });    }    render() {      return (        <div>          <p>Count: {this.state.count}</p>          <button onClick={this.incrementCount}>Increment</button>        </div>      );    }  } | import React, { useState } from 'react';  function MyComponent() {    const [count, setCount] = useState(0);    const incrementCount = () => {      setCount(count + 1);    }    return (      <div>        <p>Count: {count}</p>        <button onClick={incrementCount}>Increment</button>      </div>    );  } | | | | CO3 | K1 |
|  | Draw flow of sequence on react lifecycle components | | | CO3 | K1 |
|  | Identify the need of state in reactjs and say how it differs from props with example.  Need for State:      State is used for managing and storing data within a component that can change over time.      State is typically used for data that the component should be able to modify or update.      Components can have their own local state, which is isolated from other components.  state is used for managing and modifying data within a component, while props are used for passing data from a parent component to a child component in a read-only manner. State is mutable, while props are immutable. | | | CO3 | K1 |
|  | Extract a simple login component with name and role, Based on the role(admin,user) necessary component can be called(Conditional Rendering).  import React, { useState } from 'react';  function App() {    const [name, setName] = useState('');    const [role, setRole] = useState('');    const [isLoggedIn, setIsLoggedIn] = useState(false);    const handleLogin = () => {      // Perform authentication logic here, e.g., check credentials      // For simplicity, we'll just set isLoggedIn to true if role and name are provided      if (name && role) {        setIsLoggedIn(true);      }    }    const handleLogout = () => {      setIsLoggedIn(false);      setName('');      setRole('');    }    return (      <div>        {isLoggedIn ? (          <div>            <p>Welcome, {name}!</p>            {role === 'admin' ? (              <AdminComponent />            ) : (              <UserComponent />            )}            <button onClick={handleLogout}>Logout</button>          </div>        ) : (          <div>            <label>              Name:              <input type="text" value={name} onChange={(e) => setName(e.target.value)} />            </label>            <br />            <label>              Role:              <select value={role} onChange={(e) => setRole(e.target.value)}>                <option value="">Select Role</option>                <option value="admin">Admin</option>                <option value="user">User</option>              </select>            </label>            <br />            <button onClick={handleLogin}>Login</button>          </div>        )}      </div>    );  }  function AdminComponent() {    return <div>This is the admin component. Special admin features are available here.</div>;  }  function UserComponent() {    return <div>This is the user component. User-specific content is displayed here.</div>;  }export default App; | | | CO2 | K2 |
|  | Create a ReactJS program to increment a counter by 5 whenever a button is clicked.  import React, { useState } from 'react';  function App() {    const [count, setCount] = useState(0);    const incrementByFive = () => {      setCount(count + 5);    };    return (      <div>        <h1>Counter: {count}</h1>        <button onClick={incrementByFive}>Increment by 5</button>      </div>    );  }export default App; | | | CO3 | K3 |
| **Part – B  (3 × 10 = 30 Marks)** | | | | | |
| **ANSWER ANY THREE QUESTIONS** | | | | | |
| 11. | i) | Illustrate following types of styling components with example  (i)Inline  (ii)Internal  (iii)External  (iv) Module based CSS  1. Inline Styling:  Inline styling is done using the style attribute directly within your JSX. It's typically used for small, specific styling needs.  import React from 'react';  function InlineStylingExample() {    const inlineStyle = {      color: 'blue',      backgroundColor: 'lightgray',      padding: '10px',    };    return (      <div style={inlineStyle}>        This is styled using inline CSS.      </div>    );}  export default InlineStylingExample;  2. Internal (or Embedded) Styling:  Internal styling is done within the component's JavaScript file using JavaScript objects, and it can be more dynamic compared to inline styling.  import React from 'react';  function InternalStylingExample() {    const dynamicColor = 'red';    const dynamicPadding = '20px';    const style = {      color: dynamicColor,      backgroundColor: 'lightgray',      padding: dynamicPadding,    };    return (      <div style={style}>        This is styled using internal CSS.      </div>  );}  export default InternalStylingExample;  3. External Styling:  External styling is done by linking an external CSS file to your component. It provides a separation of concerns between styling and component logic.  // styles.css  .myComponent {    color: green;    background-color: lightgray;    padding: 15px;  }  // MyComponent.js  import React from 'react';  import './styles.css'; // Import the external CSS file  function ExternalStylingExample() {    return (      <div className="myComponent">        This is styled using external CSS.      </div>  );}  export default ExternalStylingExample;  4. Module-Based CSS (CSS Modules):  CSS Modules are a way to locally scope your CSS styles by importing them directly into your JavaScript file. It helps prevent style conflicts and provides a more modular approach.  // styles.module.css  .myComponent {    color: purple;    background-color: lightgray;    padding: 10px;  }  // MyComponent.js  import React from 'react';  import styles from './styles.module.css'; // Import the CSS module  function CSSModuleStylingExample() {    return (      <div className={styles.myComponent}>        This is styled using CSS Modules.      </div>    );  }export default CSSModuleStylingExample; | (Each 2.5 Mark) | CO2 | K2 |
| 12. | i) | Build a quiz application using ReactJS for an online learning platform. The application should allow users to take quizzes on various topics. Each quiz consists of multiple-choice questions, and users can select answers for each question. After completing the quiz, users should see their score.  Each question will have the following structure:  {    "id": 1,    "question": "What is the capital of France?",    "options": ["Berlin", "Madrid", "Paris", "Rome"],    "correctAnswer": "Paris"  }  import React, { useState } from 'react';  const quizData = [    {      id: 1,      question: 'What is the capital of France?',      options: ['Berlin', 'Madrid', 'Paris', 'Rome'],      correctAnswer: 'Paris',    },      // Add more questions here  ];  function App() {    const [currentQuestion, setCurrentQuestion] = useState(0);    const [userAnswers, setUserAnswers] = useState([]);    const [score, setScore] = useState(0);    const [showScore, setShowScore] = useState(false);    const handleAnswerClick = (selectedAnswer) => {      const isCorrect = selectedAnswer === quizData[currentQuestion].correctAnswer;      if (isCorrect) {        setScore(score + 1);      }    setUserAnswers([...userAnswers, { questionId: currentQuestion, answer: selectedAnswer }]);      const nextQuestion = currentQuestion + 1;      if (nextQuestion < quizData.length) {        setCurrentQuestion(nextQuestion);      } else {        setShowScore(true);      }    };    const handleRestartQuiz = () => {      setCurrentQuestion(0);      setUserAnswers([]);      setScore(0);      setShowScore(false);    };  return (    <div className="App">        {showScore ? (        <div>            <h2>Your Score: {score}/{quizData.length}</h2>            <button onClick={handleRestartQuiz}>Restart Quiz</button>          </div>        ) : (          <div>          <h1>Quiz App</h1>            <h2>Question {currentQuestion + 1}/{quizData.length}</h2>            <p>{quizData[currentQuestion].question}</p>            <div>              {quizData[currentQuestion].options.map((option, index) => (                <button                  key={index}                  onClick={() => handleAnswerClick(option)}              >                  {option}              </button>              ))}            </div>        </div>        )}    </div>  );  }export default App; | 3    3    3    1 | CO3 | K3 |
| 13. | i)  ii) | Demonstrate extracting components with probs with suitable examples.  import React from 'react';  function Product(props) {    return (      <div className="product">        <h2>{props.name}</h2>        <p>Price: ${props.price}</p>        <p>Category: {props.category}</p>      </div>    );  }export default Product;  import React from 'react';  import Product from './Product';  function ProductList() {    const products = [      { name: 'Product A', price: 25.99, category: 'Electronics' },      { name: 'Product B', price: 19.99, category: 'Clothing' },      { name: 'Product C', price: 9.99, category: 'Toys' },    ];    return (      <div>        <h1>Product List</h1>        {products.map((product, index) => (          <Product            key={index}            name={product.name}            price={product.price}            category={product.category}          />        )}      </div>  );  }export default ProductList;  **Write a short note on props, state initialization and update with examples.**  Props and State in React:  Props (Properties): Props are a mechanism for passing data from a parent component to a child component in React. They are read-only and allow you to configure and customize child components. Props are essential for component composition and reusability.  State: State is used to store and manage data that can change within a component. Unlike props, state is mutable and is primarily used for handling user interactions, managing component-specific data, and triggering re-renders when it changes. State is specific to a component and is not meant to be shared with other components.  Example | 2    2    1    3    2 | CO3  CO3 | K2  K2 |
| 14. | i) | Give a brief account on React Lifecycle with Application  ReactJS component lifecycle:     1. **Initialization Phase:**    * constructor(props): Initializes component state and binds event handlers.    * static getDerivedStateFromProps(props, state): Sets state based on incoming props (rarely used). 2. **Mounting Phase:**    * render(): Renders the component's UI.    * componentDidMount(): Executes after component is inserted into the DOM; used for data fetching or setup. 3. **Updating Phase:**    * static getDerivedStateFromProps(props, state): Reinvoked if new props arrive.    * shouldComponentUpdate(nextProps, nextState): Decides if component should re-render (for optimization).    * render(): Re-renders the updated UI.    * getSnapshotBeforeUpdate(prevProps, prevState): Captures information before updating DOM.    * componentDidUpdate(prevProps, prevState, snapshot): Executes after component is updated; used for side effects. 4. **Unmounting Phase:**    * componentWillUnmount(): Cleans up resources (e.g., event listeners) before component is removed from the DOM.   Example Application RealTime Clock:  import React, { Component } from 'react';  class App extends Component {    constructor(props) {    super(props);      this.state = { time: new Date() };    }  componentDidMount() {      this.timerID = setInterval(() => this.tick(), 1000);    }  componentWillUnmount() {      clearInterval(this.timerID);  }    tick() {    this.setState({ time: new Date() });    }  render() {  return (        <div>        <h1>Real-Time Clock</h1>  <p>Current Time: {this.state.time.toLocaleTimeString()}</p>      </div>    );  }  }export default App; | 3    3    4 | CO3 | K3 |