React Hooks Assignment Answers

1.

```
// ThemeContext.js
import React, { createContext, useContext, useState } from
'react';


const ThemeContext = createContext();


export const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');


  const toggleTheme = () => {
    setTheme((prevTheme) => (prevTheme === 'light' ? 'dark' :
'light'));
  };


  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};


export const useTheme = () => useContext(ThemeContext);


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

```
import { ThemeProvider, useTheme } from './ThemeContext';

const ThemedComponent = () => {
  const { theme, toggleTheme } = useTheme();

  return (
    <div style={{ background: theme === 'light' ? '#fff' : '#333',
color: theme === 'light' ? '#333' : '#fff' }}>
      <h1>Themed Component</h1>
      <button onClick={toggleTheme}>Toggle Theme</button>
    </div>
  );
};

const App = () => {
  return (
    <ThemeProvider>
      <ThemedComponent />
    </ThemeProvider>
  );
};

export default App;
```

2.

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

const FormValidation = () => {
  const nameRef = useRef();
  const emailRef = useRef();
  const passwordRef = useRef();
  const [errors, setErrors] = useState({});

  const validateForm = () => {
    const errors = {};

    // Validation logic: Check if the fields are not empty
    if (!nameRef.current.value.trim()) {
      errors.name = 'Name is required';
    }

    if (!emailRef.current.value.trim()) {
      errors.email = 'Email is required';
    } else if (!/\S+@\S+\.\S+/.test(emailRef.current.value)) {
      errors.email = 'Invalid email format';
    }

    if (!passwordRef.current.value.trim()) {
      errors.password = 'Password is required';
    }

    setErrors(errors);
```

```
  // Return true if there are no errors, indicating a valid form
  return Object.keys(errors).length === 0;
};

const handleSubmit = (e) => {
  e.preventDefault();

  // Validate the form before submission
  const isValid = validateForm();

  if (isValid) {
    // Form submission logic if the form is valid
    console.log('Form submitted successfully!');
  } else {
    console.log('Form contains errors. Please correct them.');
  }
};

return (
  <form onSubmit={handleSubmit}>
    <label>Name:</label>
    <input type="text" ref={nameRef} />
    {errors.name && <p>{errors.name}</p>}

    <label>Email:</label>
    <input type="email" ref={emailRef} />
    {errors.email && <p>{errors.email}</p>}
```

```
      <label>Password:</label>
      <input type="password" ref={passwordRef} />
      {errors.password && <p>{errors.password}</p>}

      <button type="submit">Submit</button>
    </form>
  );
};

export default FormValidation;
```

3.

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

```
const ItemList = ({ items }) => {
  const renderListItem = useCallback((item) => {
    // Render logic for each item
    return <li key={item.id}>{item.name}</li>;
  }, []);

  return (
    <ul>
      {items.map((item) => renderListItem(item))}
    </ul>
  );
};

const App = () => {
  const [items, setItems] = useState(/* an array of items */);

  // Update items state logic

  return <ItemList items={items} />;
};

export default App;
```

4.

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

const Calculator = () => {
```

```
  const [num1, setNum1] = useState(0);
  const [num2, setNum2] = useState(0);

  const result = useMemo(() => {
    console.log('Performing expensive calculation...');
    return num1 + num2; // Example of an expensive calculation
  }, [num1, num2]);

  return (
    <div>
      <input type="number" value={num1} onChange={(e) =>
setNum1(Number(e.target.value))} />
      <input type="number" value={num2} onChange={(e) =>
setNum2(Number(e.target.value))} />
      <p>Result: {result}</p>
    </div>
  );
};

export default Calculator;
```

5.

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

const DataFetchingComponent = () => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
```

```
  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await
fetch('https://api.example.com/data');
        const result = await response.json();
        setData(result);
      } catch (error) {
        setError('Error fetching data');
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  return (
    <div>
      {loading && <p>Loading...</p>}
      {error && <p>{error}</p>}
      {data && (
        <div>
          <h1>Data</h1>
          {/* Display data on the UI */}
          {/* Example: <p>{data.title}</p> */}
        </div>
```

```
      )}
    </div>
  );
};

export default DataFetchingComponent;
```