React Hooks Assignment Answers

1. UseState Hook

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

const CounterComponent = () => {
  const [counter, setCounter] = useState(0);

  const handleIncrement = () => {
    setCounter(counter + 1);
  };

  return (
    <div>
      <button onClick={handleIncrement}>Increment</button>
      <p>Counter: {counter}</p>
    </div>
  );
};

export default CounterComponent;
```

2. UseEffect Hook

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

const DataFetchingComponent = () => {
  const [data, setData] = 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) {
        console.error('Error fetching data: ', error);
      }
    };

    fetchData();
  }, []);

  return (
    <div>
      {data ? <p>{JSON.stringify(data)}</p> : <p>Loading...</p>}
    </div>
  );
};

export default DataFetchingComponent;
```

3. UseReducer Hook

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

const initialState = {
  items: [],
};

const reducer = (state, action) => {
  switch (action.type) {
    case 'ADD_ITEM':
      return {
        ...state,
        items: [...state.items, action.payload],
      };
    case 'REMOVE_ITEM':
      return {
        ...state,
        items: state.items.filter(item => item !== action.payload),
      };
    default:
      return state;
  }
};

const ShoppingCartComponent = () => {
  const [state, dispatch] = useReducer(reducer, initialState);

  const addItem = item => {
    dispatch({ type: 'ADD_ITEM', payload: item });
```

```
  };

  const removeItem = item => {
    dispatch({ type: 'REMOVE_ITEM', payload: item });
  };

  return (
    <div>
      <h2>Shopping Cart</h2>
      <button onClick={() => addItem('Item A')}>Add Item A</button>
      <button onClick={() => addItem('Item B')}>Add Item B</button>
      <div>
        {state.items.map(item => (
          <div key={item}>
            {item}
            <button onClick={() => removeItem(item)}>Remove</button>
          </div>
        ))}
      </div>
    </div>
  );
};

export default ShoppingCartComponent;
```

4. UseContext Hook

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

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
  const [data, setData] = useState('Example Data');

  return (
    <AppContext.Provider value={{ data, setData }}>
      {children}
    </AppContext.Provider>
  );
};

// ChildComponent.js
import React, { useContext } from 'react';
import { AppContext } from './AppContext';

const ChildComponent = () => {
  const { data } = useContext(AppContext);
  return (
    <div>
      <p>Data from Context: {data}</p>
    </div>
  );
};
export default ChildComponent;
```

5. UseRef Hook

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

const FormComponent = () => {
  const inputRef = useRef(null);

  const handleFocus = () => {
    inputRef.current.focus();
  };

  const handleClear = () => {
    inputRef.current.value = '';
  };

  return (
    <div>
      <input ref={inputRef} type="text" />
      <button onClick={handleFocus}>Focus Input</button>
      <button onClick={handleClear}>Clear Input</button>
    </div>
  );
};

export default FormComponent;
```

6. UseMemo Hook

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

const ExpensiveComponent = () => {
  const [count, setCount] = useState(0);

  const expensiveComputation = useMemo(() => {
    let result = 0;
    for (let i = 0; i < 1000000; i++) {
      result += i;
    }
    return result;
  }, [count]);

  return (
    <div>
      <p>Result: {expensiveComputation}</p>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
    </div>
  );
};

export default ExpensiveComponent;
```

7. UseCallback Hook

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

const ParentComponent = () => {
  const [count, setCount] = useState(0);

  const callbackFunction = useCallback(() => {
    // Perform some operation
    console.log('Callback function called');
  }, []);

  return (
    <div>
      <ChildComponent callback={callbackFunction} />
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
    </div>
  );
};

const ChildComponent = ({ callback }) => {
  // Some child component rendering
  return (
    <div>
      <button onClick={callback}>Invoke Callback</button>
    </div>
  );
};

export default ParentComponent;
```

8. Custom Hook

```
// useCustomHook.js
import { useState } from 'react';

const useCustomHook = (initialValue) => {
  const [value, setValue] = useState(initialValue);

  const handleUpdateValue = (newValue) => {
    setValue(newValue);
  };

  return { value, handleUpdateValue };
};

export default useCustomHook;

// ComponentUsingCustomHook.js
import React from 'react';
import useCustomHook from './useCustomHook';

const ComponentUsingCustomHook = () => {
  const { value, handleUpdateValue } = useCustomHook('Initial Value');

  return (
    <div>
      <p>Value: {value}</p>
      <button onClick={() => handleUpdateValue('New Value')}>
        Update Value
      </button>
```

```
    </div>
  );
};

export default ComponentUsingCustomHook;
```