Ejercicios 6 y 7

`Nombre : Juan Carlos Jara` `Curso: 3 nivel` `Materia: Aplicaciones Móviles` `Tema: Comunicacion de componentes y los ciclos de vida`

1.Este es un componente de React llamado `LeapYearChecker` que se encarga de verificar si un año dado es bisiesto o no. Veamos el código línea por línea:

1. javascript

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

   Importa la biblioteca React y la función `useState` de React para manejar el estado local en el componente.
2. javascript

   `function LeapYearChecker() {`

   Define una función de componente de React llamada `LeapYearChecker`.
3. javascript

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

   Utiliza el hook `useState` para definir dos estados locales en el componente: `year` para almacenar el año ingresado por el usuario y `result` para almacenar el resultado de la verificación.
4. javascript

   `const isLeap = (year: number) => { return year % 4 === 0 && (year % 100 !== 0 || year % 400 === 0); };`

   Define una función `isLeap` que toma un año como argumento y devuelve `true` si es bisiesto según la regla estándar (divisible por 4 pero no por 100 a menos que también sea divisible por 400).
5. javascript

   `const checkLeapYear = () => { const parsedYear = parseInt(year, 10); if (!isNaN(parsedYear)) { const leapResult = isLeap(parsedYear); setResult(`${parsedYear} is${leapResult ? '' : ' not'} a leap year.`); } else { setResult('Please enter a valid year.'); } };`

Define una función `checkLeapYear` que se llama cuando se hace clic en el botón "Check". Convierte el año ingresado en un número entero (`parsedYear`) y luego verifica si es un año válido. Si es válido, determina si es bisiesto usando la función `isLeap` y actualiza el estado `result` con el mensaje correspondiente.

6. javascript

```
return ( <div> <h2>Leap Year Checker</h2> <input type="text" value={year} onChange={(e) => setYear(e.target.value)} placeholder="Enter a year" /> <button onClick={checkLeapYear}>Check</button> <p>{result}</p> </div> );
```

Devuelve la interfaz de usuario del componente, que incluye un encabezado, un campo de entrada para el año, un botón para verificar si es bisiesto y un párrafo para mostrar el resultado.

7. javascript

```
export default LeapYearChecker;
```

Exporta el componente `LeapYearChecker` para que pueda ser importado y utilizado en otros archivos de React.

2.Este es un componente de React llamado `DNAToRNAConverter` que convierte una secuencia de ADN ingresada por el usuario en su equivalente ARN. Veamos el código línea por línea:

1. javascript

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

Importa la biblioteca React y la función `useState` de React para manejar el estado local en el componente.

2. javascript

```
function DNAToRNAConverter() {
```

Define una función de componente de React llamada `DNAToRNAConverter`.

3. javascript

```
const [dnaInput, setDnaInput] = useState(''); const [rnaOutput, setRnaOutput] = useState(''); const [errorMessage, setErrorMessage] = useState('');
```

Utiliza el hook `useState` para definir tres estados locales en el componente: `dnaInput` para almacenar la secuencia de ADN ingresada por el usuario, `rnaOutput` para almacenar el resultado de la conversión a ARN, y `errorMessage` para mostrar mensajes de error.

4. javascript

```
const convertLetter = (dnaLetter: any) => { switch (dnaLetter) { case "A": return "U"; case "C": return "G"; case "G": return "C"; case "T": return "A"; default: throw new Error("Invalid input DNA."); } };
```

Define una función `convertLetter` que toma una letra de ADN como argumento y la convierte en su equivalente de ARN según las reglas de correspondencia (A -> U, C -> G, G -> C, T -> A). Si la letra de ADN no es ninguna de estas, se lanza un error.

5. javascript

```
const toRna = (dnaString: string) => { const rnaString = dnaString .split("") .map((letter) => convertLetter(letter)) .join(""); return rnaString; };
```

Define una función `toRna` que toma una cadena de ADN y la convierte en su equivalente de ARN utilizando la función `convertLetter` para cada letra de la cadena.

6. javascript

```
const handleConvert = () => { try { const rnaResult = toRna(dnaInput.toUpperCase()); setRnaOutput(rnaResult); setErrorMessage(''); } catch (error) { setRnaOutput(''); setErrorMessage(error.message); } };
```

Define una función `handleConvert` que se llama cuando se hace clic en el botón "Convert". Convierte la cadena de ADN ingresada en mayúsculas y la pasa a la función `toRna`. Si la conversión es exitosa, actualiza el estado `rnaOutput` con el resultado y borra cualquier mensaje de error. Si ocurre un error durante la conversión, establece `rnaOutput` en vacío y muestra el mensaje de error correspondiente.

7. javascript

```
return ( <div> <h2>DNA to RNA Converter</h2> <input type="text" value={dnaInput} onChange={(e) => setDnaInput(e.target.value)} placeholder="Enter DNA sequence" /> <button onClick={handleConvert}>Convert</button> {rnaOutput && <p>RNA Output: {rnaOutput}</p>} {errorMessage && <p style={{ color: 'red' }}>{errorMessage}</p>} </div> );
```

Devuelve la interfaz de usuario del componente, que incluye un encabezado, un campo de entrada para la secuencia de ADN, un botón para realizar la conversión, y muestra el resultado de la conversión a ARN (`rnaOutput`) y cualquier mensaje de error (`errorMessage`) que pueda ocurrir.

8. javascript

```
export default DNAToRNAConverter;
```

Exporta el componente `DNAToRNAConverter` para que pueda ser importado y utilizado en otros archivos de React.

En resumen, este componente de React permite al usuario ingresar una secuencia de ADN y la convierte en su equivalente de ARN al hacer clic en un botón. También maneja errores si la entrada de ADN es inválida.

Enlace de YouTube
https://youtu.be/KCS4vV5Dfds