1.

Question 1

When using a key for your list items, what’s the best general choice?

Using an ID generated by a library that guarantees no duplications.

Using an ID coming from the data, that points to the database ID.

Using an index.

2.

Question 2

Imagine you have a specification for rendering the following list item:

<li>Ice cream - 200 cal</li>, where the name and the number of calories are coming as dynamic data. Select all the options that would correctly render the desired output:

<li>{Ice cream - 200 {item.cal}}</li>

<li>{item.name} - {item.cal} cal</li>

<li>{`${item.name} - ${item.cal} cal`}</li>

3.

Question 3

Let’s suppose you have a list of two items and a new item is added to the list. From the point of view of the React diffing algorithm, what’s the most optimal position for the new element added? Select all that apply

The new element added at the beginning.

The new element added in the 2nd position, in between the existing list items.

The new element added at the end.

4.

Question 4

What are controlled components?

A set of components whose internal state is controlled by the DOM.

A set of components whose internal state is controlled by React.

A set of components whose value is determined by React refs.

5.

Question 5

What are the features you can still achieve with uncontrolled components? Select all that apply

One time value retrieval on submit using a React ref.

Validation on submit.

Conditionally disabling the submit button.

6.

Question 6

When creating an API for context consumers via useContext, what’s the argument you have to provide to the useContext call?

The Context.Provider component.

The Context object obtained via the createContext call.

The React state that defines the global state to be injected.

7.

Question 7

Imagine the below component structure, where all components ComponentA, ComponentB and ComponentC are simple presentational components that hold no props or state:

const App = () => {

return(

<AppContext.Provider>

<ComponentA />

</AppContext.Provider>

);

};

const ComponentA = React.memo(() => <ComponentB />);

const ComponentB = () => <ComponentC />;

If the App component re-rendered for whatever reason, what would be the sequence of component re-renders that would take place?

App -> ComponentA -> ComponentB -> ComponentC.

App -> ComponentB -> ComponentC

App

8.

Question 8

Even though props and state are inherently different, what are areas where they overlap? Select all that apply.

Both props and state are plain JS objects.

Both props and state changes trigger a render update.

Both props and state are deterministic, meaning that your component always generates the same output for the same combination of props and state.

9.

Question 9

When defining a JavaScript object as a piece of local React state that will be injected as context, what is the specific React hook that allows you to keep the same object reference in memory and prevent unnecessary re-renders if the object itself hasn’t changed any values?

useCallback

useMemo

useRef

10.

Question 10

What are some possible examples of application features that are well suited to be defined as React context? Select all that apply

The application theme.

Locale preferences.

The current authenticated user.

The value of a text input.