**Question 1**

An HTML widget for a children's game represents a clickable card row. Initially, all cards are face down. Whenever a card is clicked, it is turned over to face up, and any card that was previously face up is turned over to face down. Only one card can be face up at one time. Cards that are face up should have the content 'up' while face down cards should have the content 'down'. When an 'up' card is clicked, it remains 'up'.

Implement the React component Cards that accepts the amount prop which defines how many cards there should be. The component should be rendered as a table element.

For example, after clicking the second cell on the Cards component, rendered as:

<Cards amount={4} />

the page should contain the following table:

<table>

  <tbody>

    <tr>

      <td>down</td>

      <td>up</td>

      <td>down</td>

      <td>down</td>

    </tr>

  </tbody>

</table>

Question 2

The Rating component consists of 5 stars. Each star is represented by a span element. The component should render to this HTML code:

<div id='rating'>

<span>\*</span>

<span>\*</span>

<span>\*</span>

<span>\*</span>

<span>\*</span>

</div>

When a span element is clicked, the active class should be added to that span element and to all span elements before it. Also, the active class should be removed from all span elements after it, if there are any.

For example, after the third span element has been clicked, the HTML code should look like this:

<div id='rating'>

<span class="active">\*</span>

<span class="active">\*</span>

<span class="active">\*</span>

<span>\*</span>

<span>\*</span>

</div>

Complete the Rating component so that it implements the logic of the HTML widget.

Question 3

A page with comments consists of a list of comments and a form for posting a new comment. This is its HTML code:

<ul id='commentList'>

</ul>

<form>

<input type='text' id='comment'/>

<input type='button' id='postComment' value='Post'/>

</form>

Write a setup function that registers a click handler and implements the following logic:

Each time a button with id postComment is clicked, a new <li> element should be added to the bottom of the list with id commentList, containing the value of the text field with the id comment.

If the text field is empty, the comment should not be posted.

The value of the text field should be cleared after the comment has been posted.

For example, after the comment "test" has been posted, the list's HTML code should look like this:

<ul id='commentList'>

<li>test</li>

</ul>

Question 4

Using only CSS, style the "share-button" link so that it uses the background image located at https://bit.ly/37iuXZy.

The share button should be 48px high and 48px wide and should show on the left hand side of the screen at all times, always 50px from the top of the page. The button should not affect the rest of the layout in any way, and should be visible even if the user scrolls the page.

Question 5

A web store uses the following React component for showing suggestions while users are typing product names:

const ProductSearch = (props) => {

let [name, setName] = React.useState();

let [suggestions, setSuggestions] = React.useState([]);

React.useEffect(() => {

if(name !== undefined) {

props.requestSuggestions(name, setSuggestions);

}

}, [name]);

const onChange = (event) => {

setName(event.target.value);

};

return (

<div>

<label htmlFor={"search"}>Product name</label>

<input onChange={onChange} id={"search"} list={"suggestions"}></input>

<datalist id={"suggestions"}>

{ suggestions.map((suggestion) => <option>{ suggestion }</option>) }

</datalist>

</div>

);

};

Select all the correct answers.

(Select all acceptable answers.)

1. If the props.requestSuggestions function invokes setSuggestions using [name, suggestions] as the second parameter of React.useEffect, the component will be rerendered after each React.useEffect call.
2. On the first component render, the value of the suggestions variable will be [].
3. If the requestSuggestions function is async and does not call the setSuggestion function immediately, it will throw an error.
4. The function passed to React.useEffect should call ReactDOM.render so that changes to the suggestions variable are visible.
5. If the requestSuggestions function calls setSuggestions with two items, they will be shown in the datalist.
6. Calling setName("10") will make the input field have 10 as its value.

Question 6

Which of the following statements are true for merging feature and master branches?

(Select all acceptable answers.)

1. GIT ensures that conflicts never happen.
2. A push command to the master branch can fail if the master branch was modified in the meantime.
3. After merging, git-blame will list only one developer for every modified source code file.
4. Before merging, we can sync changes from the master branch to the feature branch.
5. Each developer can have their own local branches and commit changes to them. These branches are not visible to other developers until the developer publishes the changes.
6. The feature branch cannot be branched further.