SAQ

1: HTML/CSS

- Why is <meta> important?
    - What are some important meta attributes?
- Explain the differences between block and inline elements.
- What's the difference between pseudo-class and pseudo-elements? How do you use them?
- What is the difference between visibility:hidden and display:none?
- What is SVG? Why is it significant? What are some real-world use cases?

2: CSS Layout

- What is the box model?
    - What happens when you set positive margins and negative margins?
    - What about negative padding?
- Could you explain the three design principles and give some examples?
- Can you explain responsive web design? What are some examples on a website?
    - What are some media query types and features that you'd use?
    - How do you create responsive images?
- What are z-indexes for? What properties must be set for z-index to work?
    - What values does it take? How are priorities determined?
- Explain flex box, flex container, and flex items.
    - How are they better than setting position and float?
    - What are some flex-container properties?
        - How do you horizontally align flex items? What about vertical alignment?
        - What property would you use to display a column?
    - What are some flex-item properties?
- Explain the difference between CSS transitions, animations, and transformations. How would you implement them?
- Why should we import CSS files in the <head> and JS files at the end of <body> or <script>?
- How do you preload CSS files?

3: JS Introduction

- Difference between ECMAScript and JS?
- What is a JS engine?
- Explain compilation (ahead of time), just in time compilation, and interpretation
- Why do we convert JS from ES6 back to ES5 in production?
- What is dynamic typing?
- What are the 7 primitive data types?
    - Difference between null and undefined?
    - How do you check a value's type?
        - null === undefined, null == undefined
        - typeof typeof (null === undefined)
        - typeof null

        - typeof undefined
- Explain immutability. What data types are immutable?
- What is the difference between type coercion and conversion?
- Explain template string literals. How do you do string interpolation (embed an expression)?
- What is the difference between == and ===?
- Explain short-circuit evaluation. Can you give an example?
    - Walk me through this expression: *var v5 = undefined || null || NaN || 0 && 'value5'*
    - *var v6 = 'value6' && '' || null || 'last'*
- Explain some reference data types.
- How many ways can you declare a function? What is the syntax for each way?
    - When can we omit () and {} for arrow functions?
        - Walk me through this statement: var a = b => b;
- Let's say I pass two variables to a function, one primitive data type and one reference data type. In that function, I assign values to the arguments directly. Do they affect the original variable that was passed in?
- How many ways can you iterate through an array? What is the syntax for each way?
- What makes a set different from an array?
- Can you explain the DOM?
- What functions would you use to select DOM elements in JS? How do you select them based on things like id, class, tag names?
- What is the syntax to register an event handler for an element?
    - In the event handler, how do I access the element that the event was dispatched to?
- What are some event types?
- Explain event propagation. What are the two phases? What comes first?
    - How would you register the event handler to trigger during capturing?
- What do you do to prevent propagation?
- Explain event delegation.
- What happens when you submit a form? How do I stop that default action?

4: JS Advanced

- What is hoisting?
    - Explain what is logged in the following:
        - console.log(x); var x=2;
        - console.log(x); let x=2;
        - console.log(x); const x=2;
    - Are functions hoisted?
- Explain the different scopes.
- Explain the difference between var, let, const.
    - How are const, let, and var hoisted?
- Explain execution contexts.
- What is the call stack or execution stack? What does it store? In what order does it execute functions?
- Explain the scope chain and lexical scoping.
- What are closures? Can you give me an example?
- Can you explain IIFE? How do you write it?
    - What were 2 use cases that we covered in lecture? (For loop var i, duplicate function names across separate .js files)
- What is currying?

- What are first-class functions?
- What's the difference between call/apply/bind?
- What are some new features that came with ES6? (Arrow fn, block scope, deconstruction, …, enhanced object literals, default function params)
    - What does the '…' mean?
        - How do I use it for spreading? For 'rest'?
        - How do I use it to combine two arrays? Or add elements to the beginning and end?
        - How do I use it to copy an object and update one of its key values?
    - Explain enhanced object literals.
    - Explain default parameters.
    - Explain array and object deconstruction.
        - What if I had an array of 3 elements and I only care about deconstructing the first and last element? Can I just deconstruct it into two variables?
- Talk about what the 'this' keyword refers to.

5: JS Async

- Explain synchronous vs asynchronous code.
    - Why do we prefer asynchronous code in JS?
- Is JS single-threaded or multi-threaded?
- Explain the event loop.
    - What data structures does it use?
    - When does it execute tasks in the callback queue?
    - Explain the order of events in the iife > setTimeout > iife.
- Explain AJAX and how it is used.
    - What is XML?
    - What are XHR objects, and how do we use them to make server requests?
- What are Promises?
    - How many states do they have? What are they?
    - What is the syntax for creating a new Promise?
    - How do I chain promises? When would I want to chain promises?
- Explain callback hell. How do we prevent this?
- Explain error handling when it comes to code with Promises and code without promises.
    - Explain try/catch/finally.
    - From where in a Promise chain can .catch() handle errors? What if it was thrown in an async function?
- What is fetch? How does it differ from XHR?
    - How do you handle errors with fetch requests?
- Explain the use cases for Promise.all() and Promise.allSettled().
    - What is their syntax? How do they differ?
- Explain the microtask queue, relative to the callback queue and macrotask queue.
- How does async/await work? How does it differ from Promises in terms of chaining asynchronous operations? Which is more preferred?

6: JS OOP

- What is OOP? What is the difference between classes and objects?

    - What are the four features of OOP? Explain each one.
- How does JS implement OOP?
    - What are prototypes?
    - How are they used for inheritance? How do classes inherit prototypes?
- What are two ways that you can define a class?
- What are two ways to create an object?
    - What's the difference between object.create and new?
- Do classes have the prototype or the __proto__ property? What about objects (instantiations of a class)?
- When should you declare a function in the prototype versus in the constructor function?
- What does hasOwnProperty() do?
- Explain the prototype chain.
- Explain the static keyword in a class.
    - How would you define one in an ES6 class vs a constructor function?
- What's the difference between implementing inheritance in constructor functions and ES6 classes?
- What are modules and why do we use them? How do we make them available in other files?

7: Data Structures & Algorithms

- Explain what data structures are.
    - What is the most basic kind that acts as the building block for other data structures?
    - What are some common data structures?
    - Which ones are linear? Which are non-linear?
- Explain the stack and some of its properties. Is it linear?
    - What are common stack methods?
- Explain the queue and some of its properties. Is it linear?
    - What are common queue methods?
- Explain linked lists. What's the difference between singly and doubly linked lists? Are they linear?
    - What are common linked list methods?
- Explain the binary tree and some of its properties. Is it linear?
    - What are the different kinds of relationships between nodes?
    - Explain an iterative vs recursive algorithm for tree traversal.

8: SASS, Bootstrap, & jQuery

- Explain what CSS preprocessors do. Why do we use them?
- What are the syntax differences between SASS & SCSS?
- How do you declare SASS variables? How about CSS variables?
- Explain nested SASS selectors. How would I reference the parent selector?
- What are partials and how do I create one? How do I import one?
    - Do partials result in a compiled CSS file?
- Explain what @mixin is and how to use it. How do I pass in parameters?
    - Explain interpolation; how do I do it?
- Explain @extend and placeholder selectors. How do I use them?
- What is the difference between @mixin and @extend?
- Explain bootstrap and the basic concepts (grid, screen size).
- Explain the !important rule. What are the different priorities?
- What are .min files, and why do we have them?

- Explain jQuery. Why do we use it?
- Explain why we execute code on document.ready events.
- What's the difference between DOM elements and jQuery objects?
- What are some jQuery selectors?
- How do we get and set an element's text, html, or value?
- How do we get and set an element's CSS style?
- How do we get and set an element's attributes?
    - How do we check the values of boolean attributes?
- How do we register and remove event handlers in jQuery?
    - What about custom events?
- How does event delegation work using jQuery?
- What are some example functions for applying animations in jQuery?
- How can I use jQuery to make AJAX requests?

9: Node Introduction

- What is Node.js? How is it different from JavaScript?
    - What are some differences between client-side and server-side JS (run in the browser vs in Node.js)?
    - Why is it that node.js files don't share global scopes, but client-side do?
- What is REPL?
- What is CommonJS?
    - Explain the difference between CommonJS and ES6 modules.
- What are some global objects?
- How are modules imported and exported?
- Name some common or useful npm third-party libraries and their use cases.
- Explain event-driven architecture in Node.js.
    - What are event emitters? What are some examples of objects that implement EventEmitter (http requests & responses, streams)?
- What is the process object in Node.js?
- What is npm?
- What is a RESTful API?
- Explain HTTP, requests, responses, and the request-response cycle.
    - What is the structure of an HTTP request (start line, header)?
    - What is the structure of a response (status line, header, body)?
- What is CORS? How do we solve this issue?
- How many HTTP methods are there? Name each one.
    - Which method do I use if I wanted data from a server?
    - What if I wanted to create a new resource or update the resource?
        - Explain the difference between POST and PUT.
- What are the 5 classes of HTTP status codes and when are they used?
    - What are some examples?
- What's the order of events that occur when you navigate to some url? (request > dns > get IP addr of server host > request page from IP > receive the page > display as web page)
- What are ports? Which are reserved for HTTP and HTTPS?
- Explain serialization vs deserialization.
- What is JSON? Explain JSON.parse() and JSON.stringify().

10: Node Express, Template Engines, Databases

- What is express? What are the basic features (or code) that you need to set up a basic server?
    - What is the structure of an app.get() or post method?
    - In a route url, what symbols are used to match query strings, request parameters, and any string?
- What is postman?
- What's the difference between static and dynamic web pages?
- What are template engines? What are some examples?
- What is SSR and CSR? How are they different in terms of requests being sent to and from the server?
    - What are advantages and disadvantages to using both?
- What is a relational DBMS, and how does it differ from non-relational DBMS?
    - Can you give examples?
- What are transactions?
- What is data modeling?
    - What are the different types of entity relationships? What do they mean?

11: Node MongoDB

- What are two methods of scaling up your server so that it can handle more work?
    - Explain vertical scaling vs horizontal scaling.
    - What are the advantages and disadvantages of each method?
- What is sharding and what issues does it solve?
    - What are the advantages (read/write throughput, overall storage, availability) and disadvantages (query overhead, complexity, infrastructure costs) to sharding?
- What are CRUD operations?
- What are connection pools? Why would you create a connection pool? In what situations would you want to create a connection pool?
- Can you explain collections, schemas, models, documents, and fields in MongoDB?
- What is mongoose?
- Can you name some schema types (number, string …)?
- Could you explain two-way and one-way embedding in MongoDB?
- What is the ref keyword in mongoose? How does the populate function work?
- Explain hashing. Given the hashed value, can you reverse the encryption?
- Explain the MVC architecture. How would you structure your server directory to look like MVC?
- Can you explain XSS? How would you prevent it?

12: Node Login, Authentication, Authorization

- What is a stateful vs stateless protocol? Which one describes the HTTP protocol?
    - What are ways for the server to figure out which user/client is sending the request?
- What are cookies?
- What are sessions?
- How do cookies and sessions work together to maintain user authentication & authorization?
- What does the request/response cycle look like between client & server when establishing cookies and sessions?

- Can you explain how middleware is used in cookie & session management, in regards to that request/response cycle?
- What are good practices for session management?
- What are advantages and disadvantages (pros/cons) of using cookies and sessions?
- What is JWT?
    - Explain the JWT structure.
    - What does the request/response cycle look like between client & server when establishing JWT auth?
        - When are JWTs created & encrypted, and when are they decoded?
- Can you explain the differences between establishing cookies & sessions vs JWT?
- What are advantages and disadvantages (pros/cons) of using JWT?
- Explain what CSRF is. Can you give a real-world example?
- What are some places that you could store the JWT on the client-side, and when would you use each one?
- Explain the idea of SSO. How might the requests and responses be structured (between all entities) if a client is logging into a website with their SSO account?
    - What are some advantages to using SSO instead of implementing login yourself?
- Explain the idea of OAuth 2.0. How does it differ from SSO?
    - Explain the workflow.
- What is passportjs? What are some use-cases for it?

13: Node Testing

- What are the different types of testing? Explain each one.
- What are some popular Node.js testing frameworks? How would you use them together to write test cases?
- What is BDD and TDD? Which one does Chai follow?
- How can you test your backend routes?
    - What should you be testing them for (response status, schema response, valid payload, latency)?
- What testing features does jest offer?
    - Explain what code coverage is.
- What are spies, stubs, and mocks?
- What is WebSocket?

14: Typescript

15: React Intro

- What are some differences between React and Angular?
- What is React strict mode?
- Explain JSX. What is correct syntax? ( ) and 1 outer element
    - How do we embed expressions (interpolation)?
    - How is JSX syntax different from HTML?
- What are components? Why is it helpful to divide a webpage into components?
    - What are the two types of components and their differences?
- What does ‘this’ refer to in class components vs functional components?
    - Why should we bind in class components?

- How do parent components send data to child components? How do child components update their parents?
    - Can you modify props? Why should it be read-only?
- How do you access props in class components? In functional components?
- What are states and how do they affect rendering?
    - Why shouldn't you mutate the state directly?
- How do you declare a state in class components?
    - How do you set a new state? Is it asynchronous or synchronous?
- What are controlled and uncontrolled components? How do they differ?
    - Which is recommended and why?
- What are change handlers? Why would you create one? How do you create one?
- What is the virtual DOM? Explain reconciliation.
    - Why is manipulating the virtual DOM generally faster than the actual DOM?
- Explain the diffing algorithm.
    - How does the key attribute improve the diffing algorithm?
- Explain the lifecycle methods.
    - In what order are the class component methods called?
- Explain shouldComponentUpdate() and pure components.

16: React Class Components

- Explain different ways of sharing state between components.
    - What is the Context API? How does pub-sub work?
    - What is prop drilling?
- What are SPAs? How is it different from non-SPA websites?
- Explain react-router-dom.
    - What are route components?
    - Difference between navlink and link components?
    - How can we pass data to route components and get their values?
- When might we want to programmatically redirect the user to another page?

17: React Functional Components

- What are hooks in React? Why do we use them?
- Explain the following hooks (what it does, what's the syntax). What are their class component equivalents?
    - useState
    - useEffect
    - useRef
    - useContext
- Explain how the useReducer hook works. What is its syntax, and what are some use-cases?
- What are guard routes? Why do we use them? What are some use-cases?
- What are custom hooks? What is their naming convention? Why do we use them? What are some use-cases?
- Explain the following hooks (what it does, what's the syntax).
    - useSelector
    - useDispatch
    - useStore

- What is React.memo? How does it differ from useMemo? Explain in terms of syntax and how they work.
- Explain useCallback (what it does, what's the syntax). What are some use-cases?

18: Redux

- What is the Flux architecture? What is Redux?
    - Why do we use redux? Why wouldn't we use Redux?
    - If Redux is used for shared state management, how does it compare to Context API, pub/sub, and props? What are the advantages and disadvantages of each?
- Explain the Redux structure (action, store, reducer). What is the data flow like when we want to update and read the Redux state?
- What are pure functions? Why do we want to avoid side effects? What are the benefits of pure functions?
- Talk about some of the principles of Redux.
- What is react-redux? What is its cycle of reading and updating states?
- What is combineReducers? When might we want to use it?
- What is a HOC? Can you name some examples?
    - What does connect do? What arguments does it expect?
- What is the point of mapState and mapDispatch?
- What is thunk? What are some use-cases?

Coding

HTML/CSS

1. Recreate this responsive UI using only HTML & CSS. The navbar should look like the first image when the screen size is >600px. When it is <=600px, it should look like the second image.

2. Given the HTML template, implement the function **reset_form()** so that when the **Reset** button is clicked, the email and password fields are empty.

```
<!DOCTYPE html>
<html>

<head>
 <script type="text/javascript" src="index.js"></script>
 <!-- <link rel="stylesheet" type="text/css" href="index.css"> -->
</head>

<body>
 <h1>Login</h1>
 <form name='loginForm' onsubmit="event.preventDefault()">
   <div>
     <input id='email' value="demo@mettl.com" />
     <input id='password' type="password" value="password" />
     <button id='btn' onclick="reset_form()">Reset</button>
     <div>
 </form>
</body>

</html>
```

3. Given the HTML template, add CSS to make the square size 240x240 px with a red background and its <p> content vertically and horizontally centered.

```
<!DOCTYPE html>
<html>

<head>
 <link rel="stylesheet" type="text/css" href="index.css" />
</head>

<body>
 <div id="square">
   <p>Text</p>
 </div>
</body>

</html>
```

JS

1. Implement a closure function named myFn(). If you call myFn(), it returns 1. Calling myFn() consecutively should return 2, 3, 4, …
2. Implement a currying function so that sum(1)(2)(3) = 6.
3. Implement the functions getAreas & calculateArea so that the following function call logs [3.14, 12, 9, 10]

```
const calculateArea = (shape, values) => {
 //TODO
}
const getAreas = (shapes, values_arr) => {
 //TODO
}

getAreas(['circle', 'rectangle', 'square', 'triangle'], [[1], [3, 4], [3], [4,
5]]).then(data => console.log(data)).catch(e=>console.log(e))
```

4. What is the output?

```
const arr = [1,2,3]
for(var i=0;i<arr.length;i++){
       setTimeout(function(){
              console.log(`index: ${i}, element is ${arr[i]}`)
       },1000)
}

How would you change the code so that the result is (before ES6 & after ES6)
index: 0, element is 1
index: 1, element is 2
index: 2, element is 3
```

5. Given a string `substr`, implement a function **getMovieTitles()** that:
    1) Queries **https://jsonmock.hackerrank.com/api/movies/search/?Title=substr**
        a) Replace substr with the input string.
    2) Initializes a string array, **titles** that stores every movie title in the response body.
    3) Sorts these titles in ascending order and returns it as the output.

```
// ex: getMovieTitles('Ironman') should return a promise resolved with the array
       ["Beat Feet: Scotty Smiley's Blind Journey to Ironman", 'Being Ironman', "Car and
Driver Presents Ivan 'The Ironman' Stewart", 'Darasingh: Ironman', 'Ironman Suit Office
Story', 'Ironman Triathlon Special', 'Ironman Triathlon Special', 'Ironman Triathlon World
Championship', 'Ironman Triathlon World Championship', 'Ironman Triathlon World
Championship', 'Ironman Triathlon World Championship', 'Ironman Triathlon World
Championship', 'Ironman World Championship', "Ivan Ironman Stewart's Super Off Road",
'Pushing the Limits: Diaries of an Ironman', 'Road to Ironman', 'The Ironman: Quitting Is
for Crybabies', 'The Road to Ironman']
```

React

1. Implement a progress bar using ReactJS & CSS. When the user presses **Move Forward**, the bar will increase by 10%. When the user presses **Move Backward**, it will decrease by 10%. When the bar reaches 100%, pressing **Move Forward** should alert the user **Maximum Reached**.

2. Implement the same progress bar with only one button, **Loading/Paused**. This button toggles the state. If the state is loading, then the progress bar will increase by 10% every second until it reaches 100%. If it is paused, it will not progress.
3. Given the HTML template, implement the react application so that when a user clicks on the **Next** button, it will display the next image in the images array. When the user clicks on **Previous** button, it will display the previous image.

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>slide image</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/react/16.13.1/umd/react.production.min.js"></scr
ipt>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/react-dom/16.13.1/umd/react-
dom.production.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-
standalone/6.26.0/babel.min.js"></script>
  </head>

  <body>
    <div id="root"></div>
    <script type="text/babel">
        const images = ["http://placekitten.com/g/600/400",
"http://placekitten.com/600/400", "https://placebear.com/600/400",
"https://placebear.com/g/600/400"];
        function App() {
            return(
                <div>
                    <button>Next</button>
                    <button>Previous</button>
                    <img style={imgStyle}/>
                </div>
            );
        }
        const rootElement = document.getElementById("root");
        ReactDOM.render(<App />, rootElement);
    </script>
```

```
  </body>
</html>
```

4. Given the sample React template, create a simple form that allows the user to enter in a first name, last name, and phone number. There should be a submit button such that when clicked, it displays the information in a list **automatically sorted by last name**. This list should also contain all the previous information that was entered.
    a. All fields are **required**; if the user has not inputted anything for a field, it should not be displayed in the list. On page load, the input fields should be **pre-populated** with the following values:
    First name = Coder
    Last name = Byte
    Phone = 8885559999

```
import React, { useState } from 'react';
import ReactDOM from 'react-dom';

const style = {
  table: {
    borderCollapse: 'collapse'
  },
  tableCell: {
    border: '1px solid gray',
    margin: 0,
    padding: '5px 10px',
    width: 'max-content',
    minWidth: '150px'
  },
  form: {
    container: {
      padding: '20px',
      border: '1px solid #F0F8FF',
      borderRadius: '15px',
      width: 'max-content',
      marginBottom: '40px'
    },
    inputs: {
      marginBottom: '5px'
    },
    submitBtn: {
      marginTop: '10px',
      padding: '10px 15px',
      border:'none',
      backgroundColor: 'lightseagreen',
      fontSize: '14px',
      borderRadius: '5px'
    }
  }
}
```

```
function PhoneBookForm({ addEntryToPhoneBook }) {
  return (
    <form onSubmit={e => { e.preventDefault() }} style={style.form.container}>
      <label>First name:</label>
      <br />
      <input
        style={style.form.inputs}
        className='userFirstname'
        name='userFirstname'
        type='text'
      />
      <br/>
      <label>Last name:</label>
      <br />
      <input
        style={style.form.inputs}
        className='userLastname'
        name='userLastname'
        type='text'
      />
      <br />
      <label>Phone:</label>
      <br />
      <input
        style={style.form.inputs}
        className='userPhone'
        name='userPhone'
        type='text'
      />
      <br/>
      <input
        style={style.form.submitBtn}
        className='submitButton'
        type='submit'
        value='Add User'
      />
    </form>
  )
}

function InformationTable(props) {
  return (
    <table style={style.table} className='informationTable'>
      <thead>
        <tr>
          <th style={style.tableCell}>First name</th>
          <th style={style.tableCell}>Last name</th>
          <th style={style.tableCell}>Phone</th>
        </tr>
      </thead>
    </table>
```

```
  );
}

function Application(props) {
  return (
    <section>
      <PhoneBookForm />
      <InformationTable />
    </section>
  );
}

ReactDOM.render(
  <Application />,
  document.getElementById('root')
);
```

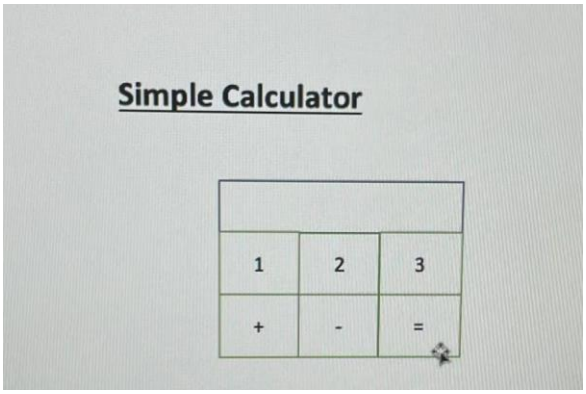
5. Given the React template, modify the application so that when you click the **Toggle** button, the favorite programming language toggles between the items in the languages array. The default value should be the first item in the array. You must use the **Context API.** Implement this class component, then convert it into a function component.

```
import React, { useState } from 'react';
import ReactDOM from "react-dom";

const language = ['JavaScript', 'Python'];

class App extends Component {
 constructor(props) {
   super(props);
 }
 render() {
   return (
     <MainSection />
   );
 }
}

class MainSection extends React.Component {
 render() {
   return (
     <div>
       <p id='favoriteLanguage'>Favorite programming language: {null}</p>
       <button id='changeFavorite'>Toggle language</button>
     </div>
   );
 }
}

ReactDOM.render(<App />, document.getElementById('root'));
```

6. Convert the following ES5 React app so that it uses ES6 syntax. List the differences.

```
var App = React.createClass({
 getInitialState: function () {
   return { name: 'world' };
 }
 render: function () {
   return <h3>Hello, {this.state.name}!</h3>;
 }
});

var App = React.createClass({
 propTypes: { name: React.PropTypes.string },
 render: function () {
   return <h3>Hello, {this.props.name}</h3>;
 }
});
```

7. Implement a Simple Calculator App with 6 buttons (1,2,3,+,-,=) and an input field. After the user presses ‘=’, all the buttons should be disabled.



Example1: When the user press ‘1’, then ‘+’, then ‘3’, the input field should display ‘1+3’, then the user press ‘=’, the input field should display ‘4’

Example2: When the user press ‘1’, then ‘1’, then ‘-’, then ‘2’, the input field should display ‘11-2, then the user press ‘=’, the input field should display ‘9’

Example3: When the user press ‘1’, then ‘-’, then ‘2’, then ‘+’, then ‘3’, the input field should display ‘1-2+3, then the user press ‘=’, the input field should display ‘2’

8. Implement a GitHub User React app. It should display the ‘id’, ‘avatar_ul’(image), ‘login’ and ‘type’ attributes from https://api.github.com/users. While waiting for the response, it should display some text **LOADING**. If we don’t get the success response, it should display some text **ERROR**.
    a. You may experience an issue when you and others connected to the same IP have submitted too many requests to this API.

(Note: For Chrome, we could change the Network settings by opening Developer Tools → Choose Network Tab → Change No throttling to Fast 3G / Slow 3G. This could simulate the slow network and you could also see ‘loading’ text)

9. Convert the following class component into a function component.

```
class Index extends Component {
 constructor(props) {
   super(props);
   this.state = {
     data: null
   };
 }
 render() {
   let listItems;
   if (this.state.data) {
     listItems = this.state.data.map((item, i) => {
       return <p key={i}>{item.name}</p>;
     });
   } else {
     listItems = <p>Loading...</p>;
   }
   return { listItems };
 }

 componentDidMount() {
   fetch('url')
     .then((res) => res.json())
     .then((json) => { this.setState({ data: json }); });
 }
}
```

10. Given the React template, implement a React application that renders the list of repositories in the following manner. You must use the given **getReactRepositories** function. Then create a HOC that will wrap the list of repositories and provide a ‘More/Less’ button that will collapse or expand the list.

- <name> - 🌟 <stars> - 🍴 <forks>
- react - 🌟 69012 - 🍴 12581
- reselect - 🌟 7291 - 🍴 214
- redux - 🌟 31705 - 🍴 6581
- recompose - 🌟 5671 - 🍴 342

```
// Expanded Mode
+-----------------------------------------------+
| <name> | 🌟 <numberOfStars> |🍴 <numberOfForks>|
+-----------------------------------------------+
| react      |     🌟 69012      |   🍴 12581    |
+-----------------------------------------------+
| reselect   |     🌟 7291       |   🍴 214      |
+-----------------------------------------------+
| redux      |     🌟 31705      |   🍴 6581     |
+-----------------------------------------------+
| recompose  |     🌟 5671       |   🍴 342      |
+-----------------------------------------------+
| See Less Button |
+-----------------+

// Collapsed Mode
+-----------------------------------------------+
| <name> | 🌟 <numberOfStars> |🍴 <numberOfForks>|
+-----------------------------------------------+
| react      |     🌟 69012      |   🍴 12581    |
+-----------------------------------------------+
| reselect   |     🌟 7291       |   🍴 214      |
+-----------------------------------------------+
| See More Button |
+-----------------+
```

```
import React, { Component } from 'react';
import { render } from 'react-dom';
import axios from 'axios';
import TaskDescription from './TaskDescription';

const SEARCH_ENDPOINT = 'https://api.github.com/search/repositories?q=react';

const getReactRepositories = () => axios.get(SEARCH_ENDPOINT)
 .then((result) => result.data.items)
 .then((repos) => repos.map(({ forks, name, stargazers_count, html_url }) => ({
   forks,
   name,
   stars: stargazers_count,
   url: html_url,
 })));
```

```
const App = () => <TaskDescription />;

render(<App />, document.getElementById('root'));
```

11. Create a sliding puzzle game. You will make a 4*4 grid that has the numbers 1-15 and one empty cell. Clicking any cell adjacent to the empty cell will swap them. If you click on the empty cell, nothing happens. The game is won when all the values in the grid are in order, and a pop up window says 'Congrats!!'

Examples:
15	3	7	2
1	4	11	13
6	10	12	14
5	*	9	8

Clicking on 10 should change the puzzle into the followings:
15	3	7	2
1	4	11	13
6	*	12	14
5	10	9	8

The winning state should look something like this, the '*' can be in any position

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	*

Node.js

1. Given a simple web application, your job is to implement middleware that console.logs a uuid, current timestamp, and the requested url path. It should also add the field **'x-request-id' = uuid** to the response header. This middleware should only be applied to /tasks.

```
const express = require('express')
const app = express()
app.get('/tasks',(req, res) => {res.send('Hello World!')})
app.get('/index',(req, res) => {res.send('Hello index!')})
app.listen(3000, () => { console.log(`Example app listening on port ${port}`)})
```