CS 300 Spring 2022 Module Exam - JavaScript

-- CS 300 Spring 2022 Module Exam - JavaScript

-- Deadline: 12:15 PM Saturday, 26th March 2022.

-- **Guidelines**:

-- There are a total of 4 questions.

-- Read the questions and hints carefully.

-- The arguments and return types of the functions should not be changed.

-- Feel free to make any helper functions where ever you need them.

-- Total Marks (40)

Submit code for answers of all the questions in a single file rollno.js.

Question#1 [8]

Create a function named, **reverseArrayInPlace**. This function *modifies* the array given as argument by reversing its elements. This function CANNOT use the standard **reverse** method. Moreover, you are NOT allowed to declare another array inside this function—you can only modify the array passed as argument, i.e., reverse array in place.

Example: If input array is [1, 2, 3], after execution of this function the array should become [3,2,1]

```
function reverseArrayInPlace (array) {


}
```

Question#2 [8]

Write a function **loopFunction** that provides functionality similar to a for loop statement. This function takes a starting **value**, a test function, an update function, and a body function as parameters. In each iteration, it first runs the **test** function on the current loop **value** and stops if that returns false. Then it calls the **body** function, giving it the current value. Finally, it calls the **update** function to create a new **value** and starts from the beginning. When defining the function, you can use a regular loop to do the actual looping.

Example:

```
loopFunction(1, (i) => i <= 3 , (i) => console.log(i), (i) => i + 1 );
```

The above call to the function prints the following:
1
2
3

```
function loopFunction( value, testFunction, bodyFunction, updateFunction) {


}
```

Question#3 [14]

The following is the behavior of the library function **Promise.all**.

Given an array of promises, Promise.all returns a promise that waits for all of the promises in the array to finish. It then returns an array of result values if all promises resolve successfully. If a promise in the array fails, the promise returned by Promise.all fails too, with the failure reason from the failing promise.

Implement a function allPromise yourselves that provides the same functionality as the library function Promise.all. You are NOT allowed to use the Promise.all function.

```
function allPromise (array) {


 }
```

Hints:
Remember that after a promise has succeeded or failed, it can't succeed or fail again, and further calls to the functions that resolve it are ignored. The function allPromise will have to call ***then*** on each of the promises in the given array. When one of them succeeds, two things need to happen. The resulting value needs to be stored in the correct position of a result array, and we must check whether this was the last pending promise and finish our own promise if it was. Also take into account the case when the input array is empty.

In order to test your function, compare your function's behavior with Promise.all library function.

Question#4 [10]

An identifier in a certain language may contain letters, $, _, and digits (0-9), but may not start with a digit. Here are additional constraints on identifiers:

a. $ and _ may not appear next to each other.
b. $ symbols may not appear next to each other, i.e., $$, $$$ etc. are not allowed in identifiers.
c. An identifier may not end at an underscore (_) symbol.

Examples of valid identifiers: ab$5c$, bc_8$

Examples of invalid identifiers: 4ad$, ab$$, ac$d_

Write a JavaScript function that takes a string as input and returns true if the parameter is a valid identifier. The function returns false otherwise. You are NOT allowed to use library functions of JavaScript string library.

```
function isIdentifier (ident) {

}
```

Hints:

Scan the string one character at a time. Think how a finite automaton works.