Web lab assignment - 7

Name : G Krishna Sai
Sec : “A2”

1.Write a function numTest that takes a number as an argument and returns a Promise that tests if the value is less than or greater than the value 20.

```
const numTest = (num) => new Promise((resolve, reject) =>{
        if (num > 20) {
                resolve(‘greater than 20’)
        }
        else {
          reject(‘less than 20’)
        }
})

numTest(22)
        .then(function(value) {
 console.log(value);
})
.catch(function(error) {
        console.log(error)
})
```

output:-

2. Write a JavaScript code to handle multiple call back functions using JavaScript promises (use promiseobject.then(onfulfilled,onrejected)).

```
let success = true;

function getUsers() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      if (success) {
        resolve([
          { username: 'krish', email: ‘krishna@test.com' },
          { username: 'jain', email: ‘arpi@test.com' },
        ]);
      } else {
        reject('Failed to the user list');
      }
```

```
    }, 1000);
  });
}

function onFulfilled(users) {
  console.log(users);
}
function onRejected(error) {
  console.log(error);
}

const promise = getUsers();
promise.then(onFulfilled, onRejected);
```

```
Failed to load resource: the server responded with a status of 404 (Not Found)
  ▼Array(2)
    ▼0:
       email: "krishna@test.com"
       username: "krish"
     ▶[[Prototype]]: Object
    ▼1:
       email: "arpi@test.com"
       username: "jain"
     ▶[[Prototype]]: Object
     length: 2
   ▶[[Prototype]]: Array(0)
```

3. Write a program to store values into a set, and to retrieve value from the set, to iterate over the set

```
var set1 = new Set();
set1.add(50);
set1.add(30);
set1.add(40);
set1.add(20);
set1.add(10);
var getEntriesArry = set1.entries();
console.log(getEntriesArry.next().value);
console.log(getEntriesArry.next().value);
console.log(getEntriesArry.next().value);
```

```
// [object Array] (2)
[50,50]

// [object Array] (2)
[30,30]

// [object Array] (2)
[40,40]

>
Console  Assets  Shortcuts
```

4. Write a program to store values into a map, and to retrieve value from the map using key, to iterate over the map

```
const ageMap = new Map([
  ‘krishn', 20],
  ['kittu', 34],
  ['abhi', 10],
  ['Sai', 9]
]);
console.log(ageMap.keys());
console.log(ageMap.values());
console.log(ageMap.entries());
ageMap.forEach((value, key) => {
   console.log(`${key} is ${value} years old!`);
});
for(const [key, value] of ageMap) {
  console.log(`${key} is ${value} years old!`);
}
```

```
▸ MapIterator
▸ MapIterator
▸ MapIterator
krishna is 20 years old!
kittu is 34 years old!
abhi is 10 years old!
Sai is 9 years old!
krishna is 20 years old!
kittu is 34 years old!
abhi is 10 years old!
Sai is 9 years old!
```

5. Write a program to iterate over a 2-dimensional array and print all the values of it

```
let chunked = [[1,2,3], [4,5,6], [7,8,9]];

for(let i = 0; i < chunked.length; i++) {

   for(let j = 0; j < chunked[i].length; j++) {

      console.log(chunked[i][j]);
   }
}
```

```
1
2
3
4
5
6
7
8
9
```

6. Write a JavaScript code to insert and remove elements from the array for the given index

```
const name = ['mahan', 'kali', 'manohar', 'sai'];
console.log(name);

console.log(name.splice(2,  0, 'phani'));
console.log(name);
```

```
> const name = ['mahan', 'kali', 'manohar', 'sai'];
  console.log(name);

  console.log(name.splice(2, 0, 'phani'));
  console.log(name);
  ▸(4) ['mahan', 'kali', 'manohar', 'sai']
  ▸[]
  ▸(5) ['mahan', 'kali', 'phani', 'manohar', 'sai']
<· undefined
>
```

Application based Problems

7. Show how map is different from object to store key value pairs with coding example and prove Maps perform better than objects in most of the scenarios involving addition and removal of keys

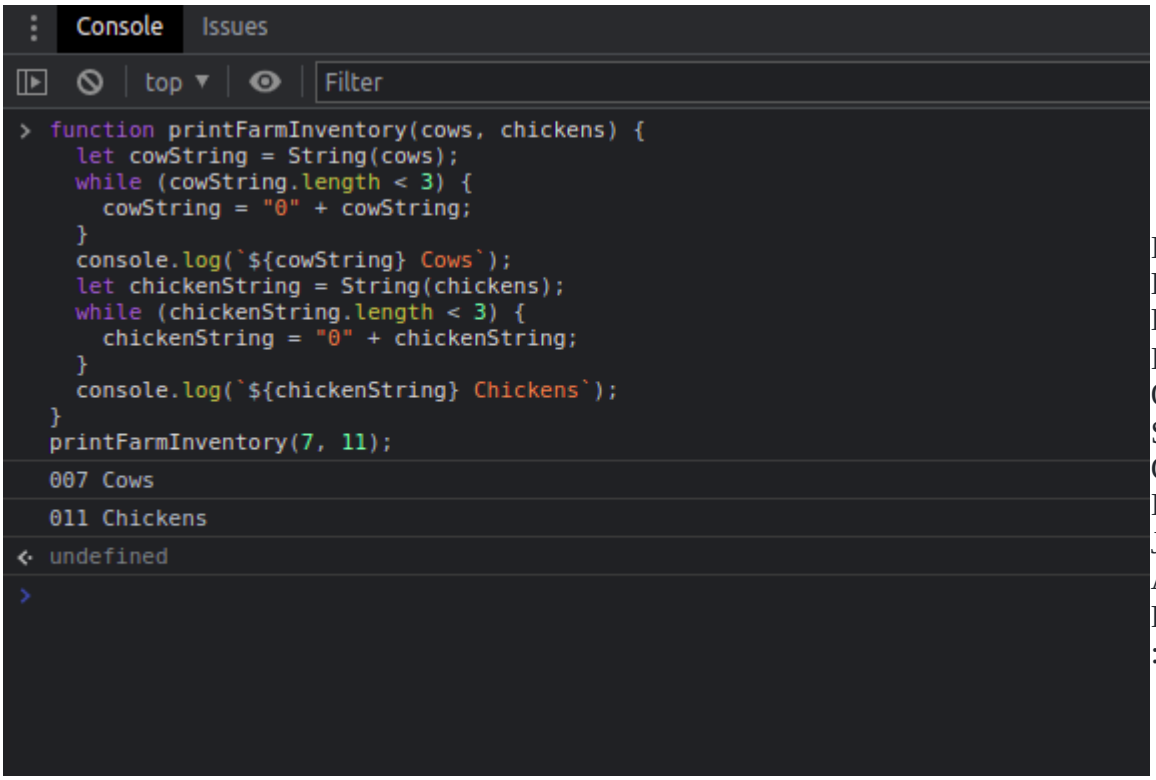
12. Write a program that prints two numbers: the numbers of cows and chickens on a farm, with the words Cows and Chickens after them and zeros padded before both numbers so that they are always three digits long using functions

Input (Function Call): printFarmInventory(7, 11);

Expected output:

007 Cows 011 Chickens

```
function printFarmInventory(cows, chickens) {
  let cowString = String(cows);
  while (cowString.length < 3) {
    cowString = "0" + cowString;
  }
  console.log(`${cowString} Cows`);
  let chickenString = String(chickens);
  while (chickenString.length < 3) {
    chickenString = "0" + chickenString;
  }
  console.log(`${chickenString} Chickens`);
}
printFarmInventory(7, 11);
```



IMPLEMENTING OF STACK IN JAVASCRIPT:

```
class Stack {
    constructor() {
        this.items = [];
    }
    add(element) {
        return this.items.push(element);
    }
    remove() {
        if(this.items.length > 0) {
            return this.items.pop();
        }
    }
    peek() {
        return this.items[this.items.length - 1];
    }
    isEmpty(){
      return this.items.length == 0;
    }
    size(){
        return this.items.length;
    }
    clear(){
        this.items = [];
    }
}

let stack = new Stack();
stack.add(1);
stack.add(2);
stack.add(4);
stack.add(8);
console.log(stack.items);

stack.remove();
console.log(stack.items);

console.log(stack.peek());

console.log(stack.isEmpty());

console.log(stack.size());

stack.clear();
console.log(stack.items);
```

```
[1, 2, 4, 8]
[1, 2, 4]
4
false
3
[]
```

IMPLEMENTATION OF QUEUE IN JAVASCRIPT

```
class Queue {
    constructor() {
        this.items = [];
    }
    enqueue(element) {
        return this.items.push(element);
    }
    dequeue() {
        if(this.items.length > 0) {
            return this.items.shift();
        }
    }
    peek() {
        return this.items[this.items.length - 1];
    }
    isEmpty(){
       return this.items.length == 0;
    }
    size(){
        return this.items.length;
    }
    clear(){
        this.items = [];
    }
}

let queue = new Queue();
queue.enqueue(1);
queue.enqueue(2);
queue.enqueue(4);
queue.enqueue(8);
console.log(queue.items);

queue.dequeue();
console.log(queue.items);

console.log(queue.peek());

console.log(queue.isEmpty());

console.log(queue.size());

queue.clear();
console.log(queue.items);
```

```
[1, 2, 4, 8]
[2, 4, 8]
8
false
3
[]
```

IMPLEMENTATION OF

LINKEDLIST USING JAVASCRIPT:

```
class Node {
	constructor(element) {
		this.element = element;
		this.next = null
	}
}
class LinkedList {
	constructor() {
		this.head = null;
		this.size = 0;
	}
	add(element) {
		var node = new Node(element);
		var current;
		if (this.head == null)
			this.head = node;
		else {
			current = this.head;
			while (current.next) {
				current = current.next;
			}
			current.next = node;
		}
		this.size++;
	}
	insertAt(element, index) {
		if (index < 0 || index > this.size)
			return console.log("Please  enter a valid index.");
		else {
			var node = new Node(element);
			var curr, prev;
			curr = this.head;
			if (index == 0) {
				node.next = this.head;
				this.head = node;
			} else {
				curr = this.head;
				var it = 0;
				while (it < index) {
					it++;
					prev = curr;
					curr = curr.next;
				}
				node.next = curr;
				prev.next = node;
			}
			this.size++;
		}
	}
	removeFrom(index) {
```

```
        if (index < 0 || index >= this.size)
                return console.log("Please  Enter a valid index");
        else {
                var curr,  prev, it = 0;
                curr = this.head;
                prev = curr;
                if (index === 0) {
                        this.head = curr.next;
                } else {
                        while (it < index) {
                                it++;
                                prev = curr;
                                curr = curr.next;
                        }
                        prev.next = curr.next;
                }
                this.size--;
                return curr.element;
        }
}
removeElement(element)  {
        var current = this.head;
        var prev = null;
        while (current != null) {
                if (current.element === element) {
                        if (prev == null) {
                                this.head = current.next;
                        } else {
                                prev.next = current.next;
                        }
                        this.size--;
                        return current.element;
                }
                prev = current;
                current = current.next;
        }
        return -1;
}
indexOf(element) {
        var count = 0;
        var current = this.head;
        while (current != null) {
                // compare each element of the list
                // with given element
                if (current.element === element)
                        return count;
                count++;
                current = current.next;
        }
        return -1;
}
isEmpty() {
```

```
		return this.size == 0;
	}
	size_of_list() {
		console.log(this.size);
	}
	printList() {
		var curr = this.head;
		var str = "";
		while (curr) {
			str += curr.element + " ";
			curr = curr.next;
		}
		console.log(str);
	}
}
var ll = new LinkedList();
console.log(ll.isEmpty());
ll.add(10);
ll.printList();
console.log(ll.size_of_list());
ll.add(20);
ll.add(30);
ll.add(40);
ll.add(50);
ll.printList();

console.log("is element removed ?" + ll.removeElement(50));
ll.printList();
console.log("Index of 40 " + ll.indexOf(40));
ll.insertAt(60, 2);
ll.printList();
console.log("is List Empty ? " + ll.isEmpty());
console.log(ll.removeFrom(3));
ll.printList();
```

```
// returns false
console.log("is List Empty ? " + ll.isEmpty());

// remove 3rd element from the list
console.log(ll.removeFrom(3));

// prints 10 20 60 40
ll.printList();
true
10
1
undefined
10 20 30 40 50
is element removed ?50
10 20 30 40
Index of 40 3
10 20 60 30 40
is List Empty ? false
30
10 20 60 40
undefined
>
```

IMPLEMENTATION OF BINARY SEARCH TREE USING JAVASCRIPT:

```
class Node{
  constructor(data) {
    this.data = data;
    this.left = null;
    this.right = null;
  };
};
class BinarySearchTree{
  constructor(){
    this.root = null;
  }
  insert(data){
    var newNode = new Node(data);
    if(this.root === null){
      this.root = newNode;
    }else{
      this.insertNode(this.root, newNode);
    };
  };
  insertNode(node, newNode){
    if(newNode.data < node.data){
      if(node.left === null){
        node.left = newNode;
      }else{
        this.insertNode(node.left, newNode);
      };
    } else {
      if(node.right === null){
        node.right = newNode;
```

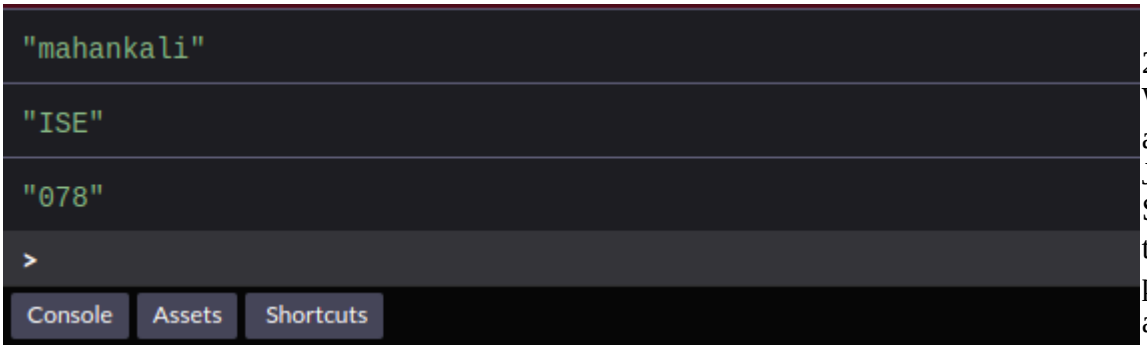
```
        }else{
          this.insertNode(node.right,newNode);
        };
      };
    };
};
const BST = new BinarySearchTree();
BST.insert(1);
BST.insert(3);
BST.insert(2);
```

JavaScript Objects and Regular Expressions

1. Write a JavaScript program to list the properties of a JavaScript object.
Sample object:
var student = {
name : "Mohan Kumar",
Dept : "ISE",
id : 056 };
Sample Output: Mohan Kumar, ISE, 056

```
let person = {
        firstname: 'mahankali',
        deptname: 'ISE',
        idnum: '078'
};

for(let key in person) {
        console.log(person[key]);
}
```



2. Write a Java Script program to search a date within a string.
Sample Input: "Albert Einstein was born in Ulm, on 14/03/1879."
Sample Output:14/03/1879.

```
var myString = "Albert Einstein was born in Ulm, on 14/03/1879.";
var myRegexp = /\d{2}[-.\/]\d{2}(?:[-.\/]\d{2}(\d{2})?)?/g;
var validDate = /(\d{1,2}[\.\/]){2,2}(\d{2,4})?/g;
myString = myRegexp.exec(myString)
myString = validDate.exec(myString[0])
console.log(myString[0]);
```

output:-

```
> var myString = "Albert Einstein was born in Ulm, on 14/03/1879.";
  var myRegexp = /\d{2}[-.\/]\d{2}(?:[-.\/]\d{2}(\d{2})?)?/g;
  var validDate = /(\d{1,2}[\.\/]){2,2}(\d{2,4})?/g;
  myString = myRegexp.exec(myString)
  myString = validDate.exec(myString[0])
  console.log(myString[0])
  14/03/1879
<· undefined
>
```

3. Write a pattern that matches e-mail addresses. Syntax: localpart@domain

Note: The local part (The text before @ symbol) contains the following ASCII characters.

Uppercase (A-Z) and lowercase (a-z) English letters.

Digits (0-9).

Characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~

Character . (dot) provided that it is not the first or last character and it will not come one after the other.

```
function valid_email(str)
{
var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)(\.\w{2,3})+$/;
if(mailformat.test(str))
{
console.log("Matches the pattern");
}
else
{
console.log("Not Matches the pattern");
}
}
valid_email('mahankali@mahankali.com
```

Error! Hyperlink reference not valid.

output:

```
> function valid_email(str)
  {
  var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)(\.\w{2,3})+$/;
  if(mailformat.test(str))
  {
  console.log("Matches the pattern");
  }
  else
  {
  console.log("Not Matches the pattern");
  }
  }

  valid_email('mahankali@mahankali.com');
  Matches the pattern
<· undefined
>
```