North South University

Department of Electrical and Computer Engineering
Bashundhara, Dhaka-1229, Bangladesh

Faculty: Dr. M. Arifur Rahman (MAh1)

Instructor: Mashnoon Islam

CSE482 Lab 3: Programming in JavaScript

Variable Declaration

- `var cse482 // scoped to immediate function body)`
- `let cse482 // scoped to immediate block. Recommended)`

Primitive Data Types

- **undefined**
- **number** (12, 12.04, Infinity, NaN)
- **string** ("CSE482", "Hello Dhaka!")
- **boolean** (true, false, [falsy: false, 0 "", undefined, NaN], [truthy: not "falsy"])
- **null**
- **bigint**
- **symbol**

Functions

```
function cse482 () {
    var cse482 = "cse482"
    console.log("Hi from function " + cse482)
}

var randomFunction = function () {
    console.log("Hi from random function!")
}

function printInputString (stringInput) {
    stringInput += " This part is concatenated from function!"
    console.log(stringInput)
}
```

```
cse482()
randomFunction()
printInputString("This is a string input!")
```

Classes and Objects

```
class Circle {
    constructor (radius) {
        this.radius = radius
    }

    perimeter () {
        return 2 * Math.PI * this.radius
    }

    area () {
        return Math.PI * Math.pow(this.radius, 2);
    }

}

var newCircle = new Circle(2)

console.log(newCircle.area())
console.log(newCircle.perimeter())
```

Another Method For Object Creation

```
var newCircle = {
    radius: 2,
    perimeter:
        function () {
            return 2 * Math.PI * this.radius
        },
    area:
        function () {
            return Math.PI * Math.pow(this.radius, 2)
        }
}

console.log(newCircle.area())
console.log(newCircle.perimeter())
```

Arrays

- Can be sparse ([100, 0, , , , 2])
- Can be polymorphic ([2, true, "CSE482", 2.009])
- Some of the many methods: **push**, **pop**, **shift**, **unshift**, **sort**, **reverse**, **length**. Look them up!

No Pointers in JavaScript

When you pass a variable (string, object, function, number, etc) to a function or an object, it is either pass-by-value or pass-by-reference. Primitive data types (string, number, etc) are passed by value, and complex data types (object, function) are passed by reference.

This Week's Task

A fragment of the source code for a Binary Search Tree (BST) in JavaScript is given down below:

```javascript
class Node {
    constructor(val){
        this.val = val
        this.left = null
        this.right = null
    }
}

class BST {
    constructor(){
        this.root = new Node(null)
    }

    insert (val) {
        this.insert_val(val, this.root)
        console.log(val + " has been inserted")
    }

    insert_val (val, node) {
        if (node.val == null) {
            node.val = val
            return
        }
        else if (val < node.val) {
            if (node.left == null)
                node.left = new Node(null)
            this.insert_val(val, node.left)
        }
        else {
            if (node.right == null)
                node.right = new Node(null)
            this.insert_val(val, node.right)
        }
    }

    print_level_order () {
        if (this.root.val == null)
            return "Empty tree"

        let visited = [],
        queue = [],
        current = this.root
        queue.push(current)

        while (queue.length) {
            current = queue.shift()
            visited.push(current.val)

            if (current.left != null)
                queue.push(current.left)
            if (current.right != null)
                queue.push(current.right)
        }
        return visited
    }

    search (val) {

    }

    print_pre_order () {

    }
    print_post_order () {

    }}
```

Create a file named **bst.js** and copy the above source code into the file. Create another file named **test_bst.html** in the same directory and insert the following code:

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript" src="bst.js"></script>
    </head>
    <body>
        <script type="text/javascript">

            const tree = new BST()
            tree.insert(20)
            tree.insert(14)
            tree.insert(57)
            tree.insert(9)
            tree.insert(19)
            tree.insert(31)
            tree.insert(62)
            tree.insert(3)
            tree.insert(11)
            tree.insert(72)

            console.log(tree.print_level_order())
        </script>
    </body>
</html>
```

Your task will be to complete the functions **search**, **print_pre_order** and **print_post_order**. The functions are described below:

1. **search**: This function will take in a value and search for its presence in the Binary Search Tree. If present, the function returns **true**, and **false** otherwise.
2. **print_pre_order** and **print_post_order** will print your Binary Search Tree in pre-order and post-order patterns respectively. Look them up if needed.