Linked Lists, Queues and Stacks

Linked List

Note: For this exercise you will need the constructor function Node:

```javascript
function Node(val, ref) {
  this.val = val;
  this.ref = ref;
}
```

Write a constructor function LinkedList that doesn't receives any argument but it initializes an attribute head with the value null.

Add a method `add` that receives a value creates a Node with the specified value. If head is null assign the new node to head. Otherwise you have to iterate through all the references until you find the last one (the one without reference). Assign the new node to the ref of that node.

Add a method `addAt` that receives a position and a value. It creates a Node with the specified value and iterates through the nodes until it finds the correct position. Set the ref of the new node to the ref of the current node. Then update the ref of the current node to the new node.

Add a method `valueAt` that receives a position and returns the value at the specified position. you will have to iterate through the nodes until you find the correct position.

Add a method `removeAt` that receives a position and removes the node at that position (you will have tu update the ref of the previous node with the ref of the node you are removing).

```javascript
const list = new LinkedList();
list.add('a');
list.add('b');
list.add('d');
list.addAt(2, 'c');

list.valueAt(0); // 'a'

list.removeAt(0);
```

Queue

Write a constructor function called `Queue` with no arguments. It should initialize three attributes:

* `head` with `null`
* `tail` with `null`
* `size` with `0`

The methods that you will implement are:

* enqueue
* dequeue
* size

```javascript
var queue = new Queue();
queue.enqueue(4);
queue.dequeue(); // returns the first element of the queue
queue.size(); // returns the size of the queue
```

Stack

Write a constructor function called Stack with no arguments. It should initialize two attributes:

* `head` with `null`
* `size` with `0`

The methods that you will implement are:

* push
* pop
* size

```javascript
var stack = new Stack();
stack.push(5);
stack.push(8);
stack.pop(); // 8
stack.pop(); // 5
stack.pop(); // null
stack.size(); // 0
```

Balanced Parenthesis (optional)

Write a function called isBalanced that receives a string and returns true if the parenthesis are balanced, false otherwise:

```javascript
isBalanced("((((()))"); // true
isBalanced("((((()))"); // false
isBalanced("())(()"); // false
```

Es posible utilizar una de las estructuras de datos que acabamos de crear