**Q7: Caching using LRU Cache**: Implement an **LRU (Least Recently Used) Cache** system using a combination of **hash maps** and **doubly linked lists** to store frequently accessed data efficiently.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>LRU Cache</title>

<style>

body { font-family: Arial, sans-serif; padding: 20px; }

input, button { padding: 10px; margin: 5px; }

pre { background-color: #f5f5f5; padding: 10px; }

</style>

</head>

<body>

<h1>LRU Cache System</h1>

<label for="key">Key:</label>

<input type="text" id="key" />

<label for="value">Value:</label>

<input type="text" id="value" />

<button onclick="setCache()">Set Cache</button>

<button onclick="getCache()">Get Cache</button>

<h2>Cache Content:</h2>

<pre id="cacheContent"></pre>

<script>

class LRUCache {

constructor(capacity) {

this.capacity = capacity;

this.cache = new Map();

}

set(key, value) {

if (this.cache.size >= this.capacity) {

this.cache.delete(this.cache.keys().next().value); // Remove least recently used

}

this.cache.set(key, value);

}

get(key) {

if (this.cache.has(key)) {

const value = this.cache.get(key);

this.cache.delete(key); // Remove and re-insert to mark as most recently used

this.cache.set(key, value);

return value;

}

return null;

}

}

const cache = new LRUCache(3);

function setCache() {

const key = document.getElementById('key').value;

const value = document.getElementById('value').value;

cache.set(key, value);

updateCacheContent();

}

function getCache() {

const key = document.getElementById('key').value;

const value = cache.get(key);

alert(value !== null ? `Value: ${value}` : 'Key not found');

}

function updateCacheContent() {

const cacheContent = JSON.stringify([...cache.cache], null, 2);

document.getElementById('cacheContent').innerText = cacheContent;

}

</script>

</body>

</html>