

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using System.Collections.Generic;
4
5 public class Program {
6     public class LRUCache {
7         public Dictionary<string, DoublyLinkedListNode> cache = new Dictionary<string,
8             DoublyLinkedListNode>();
9         public int maxSize;
10        public int currentSize = 0;
11        public DoublyLinkedList listOfMostRecent = new DoublyLinkedList();
12
13        public LRUCache(int maxSize) {
14            this.maxSize = maxSize > 1 ? maxSize : 1;
15        }
16
17        // O(1) time | O(1) space
18        public void InsertKeyValuePair(string key, int value) {
19            if (!cache.ContainsKey(key)) {
20                if (currentSize == maxSize) {
21                    evictLeastRecent();
22                } else {
23                    currentSize++;
24                }
25                cache.Add(key, new DoublyLinkedListNode(key, value));
26            } else {
27                replaceKey(key, value);
28            }
29            updateMostRecent(cache[key]);
30        }
31
32        // O(1) time | O(1) space
33        public LRUResult GetValueFromKey(string key) {
34            if (!cache.ContainsKey(key)) {
35                return new LRUResult(false, -1);
36            }
37            updateMostRecent(cache[key]);
38            return new LRUResult(true, cache[key].value);
39        }
40
41        // O(1) time | O(1) space
42        public string GetMostRecentKey() {
43            return listOfMostRecent.head.key;
44        }
45
46        public void evictLeastRecent() {
47            string keyToRemove = listOfMostRecent.tail.key;
48            listOfMostRecent.removeTail();
49            cache.Remove(keyToRemove);
50        }
51
52        public void updateMostRecent(DoublyLinkedListNode node) {
53            listOfMostRecent.setHeadTo(node);
54        }
55
56        public void replaceKey(string key, int value) {
57            if (!this.cache.ContainsKey(key)) {
58                return;
59            }
60            cache[key].value = value;
61        }
62    }
63
64    public class DoublyLinkedList {
65        public DoublyLinkedListNode head = null;
66        public DoublyLinkedListNode tail = null;
67
68        public void setHeadTo(DoublyLinkedListNode node) {
69            if (head == node) {
70                return;
71            } else if (head == null) {
72                head = node;
73                tail = node;
74            } else if (head == tail) {
75                tail.prev = node;
76                head = node;
77                head.next = tail;
78            } else {
79                if (tail == node) {
80                    removeTail();
81                }
82                node.removeBindings();
83                head.prev = node;
84                node.next = head;
85                head = node;
86            }
87        }
88
89        public void removeTail() {
90            if (tail == null) {
91                return;
92            }
93            if (tail == head) {
94                head = null;
95                tail = null;
96                return;
97            }
98            tail = tail.prev;
99            tail.next = null;
100        }
101    }
102
103    public class DoublyLinkedListNode {
104        public string key;
105        public int value;
106        public DoublyLinkedListNode prev = null;
107        public DoublyLinkedListNode next = null;
108
109        public DoublyLinkedListNode(string key, int value) {
110            this.key = key;
111            this.value = value;
112        }
113
114        public void removeBindings() {
115            if (prev != null) {
```

```
116         prev.next = next;
117     }
118     if (next != null) {
119         next.prev = prev;
120     }
121     prev = null;
122     next = null;
123 }
124 }
125
126 public class LRUResult {
127     public bool found;
128     public int value;
129
130     public LRUResult(bool found, int value) {
131         this.found = found;
132         this.value = value;
133     }
134 }
135 }
136
```