Our Solution(s) Run Code

Solution 1

```go
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

package main

type LRUCache struct {
  index            map[string]*DoublyLinkedListNode
  maxSize          int
  currentSize      int
  listOfMostRecent *DoublyLinkedList
}

func NewLRUCache(size int) *LRUCache {
  lru := &LRUCache{
    index:            map[string]*DoublyLinkedListNode{},
    maxSize:          size,
    currentSize:      0,
    listOfMostRecent: &DoublyLinkedList{},
  }
  if lru.maxSize < 1 {
    lru.maxSize = 1
  }
  return lru
}

// O(1) time | O(1) space
func (cache *LRUCache) InsertKeyValuePair(key string, value int) {
  if _, found := cache.index[key]; !found {
    if cache.currentSize == cache.maxSize {
      cache.evictLeastRecent()
    } else {
      cache.currentSize += 1
    }
    cache.index[key] = &DoublyLinkedListNode{
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

// Do not edit the class below except for the insertKeyValuePair,
// getValueFromKey, and getMostRecentKey methods. Feel free
// to add new properties and methods to the class.
type LRUCache struct {
  maxSize int
  // Add fields here.
}

func NewLRUCache(size int) *LRUCache {
  // Write your code here.
  return nil
}

func (cache *LRUCache) InsertKeyValuePair(key string, value int) {
  // Write your code here.
}

// The second return value indicates whether or not the key was found
// in the cache.
func (cache *LRUCache) GetValueFromKey(key string) (int, bool) {
  // Write your code here.
  return -1, false
}

// The second return value is false if the cache is empty.
func (cache *LRUCache) GetMostRecentKey() (string, bool) {
  // Write your code here.
  return "", false
}

```

Our Tests

Custom Output Submit Code

```
var letterMap = map[string]int{
    'a': 0,
    'b': 1,
    'c': 2,
    'd': 3,
    'e': 4,
    'f': 5,
    'g': 6,
    'h': 7,
    'i': 8,
    'j': 9,
}

var letters = []string{"a", "b", "c", "d", "e", "f", "g", "h", "i", "
```

Run or submit code when you're ready.