Assignment 2.1

```
package driver;
import java.util.HashMap;

class Node {
	int key;
	int value;
	Node ancestor;
	Node successor;

	public Node(int key, int value) {
		this.key = key;
		this.value = value;
	}
}

public class LRUCache {
	private HashMap<Integer, Node> map;
	private int capacity, count;
	private Node head, tail;
	public LRUCache(int capacity) {
		this.capacity = capacity;
		map = new HashMap<>();
		head = new Node(0,0);
		tail = new Node(0,0);
		head.successor = tail;
		head.ancestor = null;
		tail.successor = null;
		tail.ancestor = head;
		count = 0;
	}

	public void removeNode(Node n) {
		n.ancestor.successor = n.successor;
		n.successor.ancestor = n.ancestor;
	}

	public void insertNode(Node n) {
		n.successor = head.successor;
		n.successor.ancestor = n;
		n.ancestor = head;
		head.successor = n;
	}

	public int get(int key) {
		if (map.get(key) != null) {
			Node n = map.get(key);
			int result = n.value;
			removeNode(n);
			insertNode(n);
			System.out.println("node found -> key: " + key + " value: " + result);
			return result;
		} else {
			System.out.println("node not found -> key: " + key);
			return -1;
		}
```

```
    }

    public void set(int key, int value) {
        System.out.println("Set key = " + key + " and value = " + value);
        if(map.get(key) != null) {                          // node exists
            Node n = map.get(key);
            n.value = value;
            removeNode(n);
            insertNode(n);
        }else {                                                             // node does not
exist, create new one
            Node n = new Node(key, value);
            map.put(key, n);
            if(count < capacity) {                          // if total amount of node is fewer than
maximum capacity, simple add new node
                count++;
                insertNode(n);
            }else {                                                 // maximum
capacity reached, remove oldest node and add newest node
                map.remove(tail.ancestor,key);
                removeNode(tail.ancestor);
                insertNode(n);
            }

        }
    }


    public static void main(String[] args) {
        System.out.println("initializing LRUCache...");
        LRUCache cache = new LRUCache(2);
        System.out.println("LRUCache initialized.");

        cache.set(1,5);
        cache.set(2,10);    //current cache: [(1,5),(2,10)]
        cache.get(2);           //current cache: [(2,10), (1,5)]
        cache.set(3,100);   //current cache: [(1,5),(3,100)]
        cache.get(3);           //since node of key = 2 has been removed, should return -1;
        cache.set(4,99);    //current cache: [(3,100),(4,99)]
        cache.get(1);           //return -1
        cache.get(2);           //return -1
        cache.get(3);
        cache.get(4);
    }
```

Assignment 2.2

View vs Stored Procedure

View is simple showcasing data stored in the database tables whereas a stored procedure is a group of statements that can be executed. A view is faster as it displays data from the tables referenced whereas a stored procedure executes sql statements.

A View:

- Does **NOT** accept parameters

- Can be used as building block in a larger query
- Can contain only one single SELECT query
- Can **NOT** perform modifications to any table
- But can (sometimes) be used as the target of an INSERT, UPDATE or DELETE statement.

A Stored Procedure:

- Accepts parameters
- Can **NOT** be used as building block in a larger query
- Can contain several statements, loops, IF ELSE, etc.
- Can perform modifications to one or several tables
- Can NOT be used as the target of an INSERT, UPDATE or DELETE statement.
-

Views	Materialized Views
Views are not stored physically on the disk	Materialized Views are stored on the disc.
View can be defined as a virtual table created as a result of the query expression	Materialized View is a physical copy, picture or snapshot of the base table
A view is always updated as the query creating View executes each time the View is used	Materialized View is updated manually or by applying triggers to it.
Slower efficiency	Materialized View responds faster because of the precomputed feature
View is just a display hence it do not require memory space	Materialized View utilizes the memory space as it stored on the disk

Conclusion: Materialized View responds faster as compared to View. But View always provides up to date information to the user.

Reference:
https://techdifferences.com/difference-between-view-and-materialized-view.html
https://stackoverflow.com/questions/5194995/what-is-the-difference-between-a-stored-procedure-and-a-view