AP Computer Science with Data Structures

# Lab: Hashing

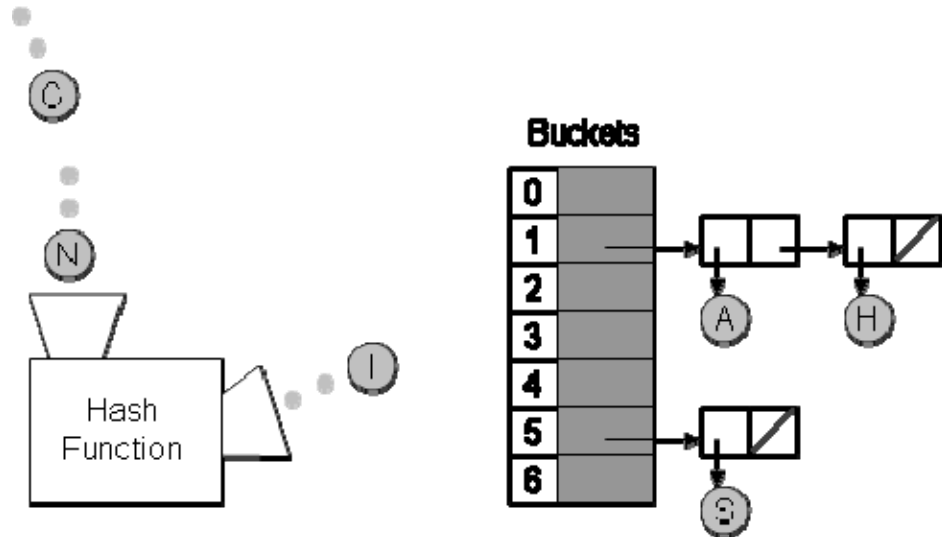
Throughout this lab, you are *REQUIRED* to document your code. Correct English usage counts as does correct and precise statements about functionality.

## *MyHashSet*

Download the file MyHashSet.java, which will imitate Java's built-in `HashSet<E>` class.

You will, of course, use hashing to implement this functionality. This class maintains an array of buckets (however many you like), each using an instance of Java's `LinkedList<E>` class to handle collisions.

Complete the helper method called `toBucketIndex`, which should take in an `Object` and return the index of the bucket where that `Object` should be found. This method should take the absolute value of the `Object`'s `hashCode`, and return its remainder when divided by the number of buckets. The result will be the object's bucket number.




Next, use your `toBucketIndex` method to complete following methods in `MyHashSet<E>`.

- `boolean contains(Object obj)`
- `boolean add(E obj)` `// if obj is not present in this set, adds obj and // returns true; otherwise returns false`
- `boolean remove(Object obj)` `// if obj is present in this set, removes obj and // returns true; otherwise returns false`

**Each of these methods must run in *O*(1) expected time.** To aid you in debugging, a `toString` method has been provided, which indicates which buckets contain which objects.Rectangle

Download Rectangle.java, and modify it so that you will be able to store rectangles in your `MyHashSet<E>`. Two `Rectangle`s will be considered equivalent whenever they share the same dimensions. For example, a 2x3 rectangle is equivalent to another 2x3, but not to one with dimensions 3x2.

Now download HashSetTester.java, and use it to demonstrate that your `MyHashSet<E>` works (or to help you debug). Demonstrate that your work passes the tester.

## *If you finish early….*

- Add an `iterator` method to your `MyHashSet` class, which should (a) run in *O*(1) time, and (b) return an `Iterator` whose methods' running times do not depend on the size of the set. Create your new class definition *inside* the `MyHashSet` class definition, so that it has access to `MyHashSet`'s private data. You may use the `IteratorTester` outside the Starting Code to test your work.

- Create a class `MyHashMap`, which implements the `Map` interface methods, using an array of `ListNode`-based linked lists, with each node containing an instance of a `MapEntry` class (pairing of key and value Objects). Demonstrate that it works. You can use the given skeletons for the `MapEntry` and `MyHashMap`. Make your life simpler by using a linked list to store the values instead of using `ListNodes`. You can use the contains methods even though it is not on the AP CS reference. (This is only for the year of the pandemic)