20.18 Program 9: Hash Table

Program 9: Hashmap

In this program, you'll be writing your own hash table.

Specifications

* Name your class MyHashTable. It will use integer keys and store values of any kind.
* Ensure it is a generic class. V is the generic type parameter of the value stored in a hash table instance. So your class is MyHashTable<V>.
* There is no limit to the number of items that can be inserted into the hash table, so ensure your hash table resizes appropriately.

Constructors

There will only be one constructor used: MyHashTable(). It will be automatically generated, as its the default constructor, but you'll may want to implement it yourself as there could be internal variable initialization that your class' implementation may need.

You are free to add more constructors if you wish, however.

Methods

Add the following methods to your class:

* void put(V value): Puts value in the hash table at the corresponding key location.
* boolean contains(V value): Returns true if value exists in the hash table and false otherwise.
* void remove(V value) throws RuntimeException: Remove value from the table. If value doesn't exist, a RuntimeException must be thrown.

Hash function

You're free to use any hash function you'd like to. **The most important thing to remember** is that you want to use value.hashCode() first, to get a key out of your value, and then you can modify it as you wish. **The next most important thing to remember** is that the hashCode() method can return *negative values*, so remember to handle them. The easiest thing to do would be to simply negate the negation and use it as a positive value.

Collision handling

You are free to use any technique you wish to use for handling collisions. ZyBooks has a lot of code resources available for all the techniques, though I believe chaining and linear probing should be the easiest to implement.

Things to look out for

What if V is an Integer? Which version of remove() would be invoked?

Hints

* The simplest hash function is the % (modulo) operator
* You could reuse contains() to implement remove() more quickly
* You can use an array of V to store your data, which means chaining will not be possible and you'll have to use a probing method, and you'll also have handle the resizing
* You can use a List<V> to store your data, easing the resizing of your hash table
* You can use a List<List<V>> or List<V>[] if you want to use chaining

Turning In Procedure

* You are required to submit the MyHashTable.java in Zybooks which will be autograded.
* The automatic grading program is very specific. If you feel you have the correct solution but are not receiving full credit, please
  + Carefully review the output -- you might need to scroll all the way to the right to find what is wrong with a particular output.
  + Verify you have the correct names for the program itself and all methods.
  + Check your calculations by hand: was there a logic error?
  + Review the requirements: did you miss a step? misinterpret a requirement?
  + If all these check out, contact the T.A. for assistance.