A multiset is a collection that extends the idea of a set to have duplicate items. It is able to keep track of the number of occurrences of each duplicate in the set.

For this question, you will implement LinkedMultiHashSet, a multiset that internally uses a resizing array and hashing (based on element's hashCodes) to make most operations run in O(1) average time. In addition to the ordinary methods for a set, you will need to implement an Iterator which returns elements in a particular order (see below and Javadoc).

You should also state the worst case time and space complexities of all your methods (in Big-O notation) in their respective Javadocs.

To gain full marks, you should ensure that:

• The internal hashtable array should start with the initial capacity given as the argument to the constructor of the class.

• The internal hashtable array should be doubled in size whenever an add operation would make the array become full. It should not reduce in size.

• Duplicate occurrences of elements should not take up additional space.

• You should use hashCode to hash objects and equals to check equality of elements.

• When a collision occurs (unequal elements which have the same hashCode), you should use linear probing to find the next position in the hashtable.

Constraints:

• Your implementation should all be inside LinkedMultiHashSet.java. You may create additional (private) inner classes however.

• You shouldn't use any additional classes from the Java Collections Framework (e.g. ArrayList, LinkedList, HashMap, HashSet). If you wish to utilise similar functionality, you should implement the needed data structures yourself.