0. Review

In the worst case, how long does it take to insert into a Linked List?
Θ(n), the insert may be at the end of the list
In the worst case, how long does it take to add an element into an Array List?
(Assuming the array has space)
Θ(1), no matter where you insert, it takes constant time in an array
In the worst case, how long does it take to retrieve an element from a Linked List?
Θ(n), same logic as above
In the worst case, how long does it take to retrieve an element from an Array?
Θ(1), same logic as above
How long does it take to resize a Linked List by one element?
O(1)
How long does it take to resize an Array by one element? (No space left in array)
O(n), have to make a new array, copy over the elements, and then add the new element

Given the above, when would we use an array over a linked list? Could you give an example?
If you know in advance how large your data structure is. When that is the case, arrays are faster than linked lists in insertion, mutation, etc. However, if the array needs to expand frequently, then things get expensive.. There are ways to amortize the cost of resizing (for example, the AList/ArrayList). Other examples include:

- things with rows/columns (like a checkers board), where we know the exact position and we know that the board is of a fixed size
- arguments to a java program (`String[] args`). Using a List wouldn't make things any easier, since the arguments to a program don't change once we start to run it.

1. HashMap Basics

Let's build a Map out of an array. Recall (a subset of) the map interface:

interface Map<K, V> {

```
  public int size();
  public void put(K key, V value);
  public boolean containsKey(K key);
  public V get(K key);
}
```

Consider this put method, which assume no collisions (and no negative hashCodes):

```
public void put(K key, V value) {
   arr[key.hashCode() % arr.length] = value;
}
```

What is the point of mod (%)? Keep indexing within the bounds of the array.

Now what are collisions, and how do we address them? (What one line changes do we have to do to the above code?) What should our array declaration look like? Assume you have a nested class Entry, where each Entry models one key-value pair.

```
class HashMap<K, V> {
    List<Entry>[] arr;

    class Entry {
        public K key;
        public V value;
        public Entry(K key, V value) {
            this.key = key;
            this.value = value;
        }
    }

    public void put(K key, V value) {
        int bucket = key.hashCode() % arr.length;
        // Make a new list if it is empty
        if (arr[bucket] == null) arr[bucket] = new LinkedList<Entry>();
        List<Entry> list = arr[bucket];
        for (Entry e : list) {
            if (e.key.equals(key)) {
                e.value = value;
                return;
            }
        }
```

```
            list.append(new Entry(key, value));
        }
    }
```

List<Entry>[] arr;
If there is already a Linked List at that index, add the key, value pair to the linked list. Otherwise, create a Linked List of size 1 containing the key value pair we want to add to the map.

How do we write get and contains key? Assume the solution for collisions you did above.
Hash the key, mod it by the array length, and iterate through the Linked List at that index to find the value associated with this key.

How would we write size? Describe in few sentences.
Iterate through array and sum the size of each linked lists.

Alternatively, initialize: private int _size = 0
_size++ when putting and not overwriting (aka, after list.append)
_size-- if you need to remove

2. HashCodes

Which of the hashCodes are invalid? Assume we are trying to hash the following class:

```
import java.util.Random;
class Point {
  private int x;
  private int y;
  private static count = 0;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
    count += 1;
  }
}
```

1.)

```
public void hashCode() {
  System.out.print(this.x + this.y);
}
```

No. Doesn’t return anything

2.)

```
public int hashCode() {
  Random randomGenerator = new Random();
  return randomGenerator.nextInt(Int);
}
```

No. Not deterministic

3.)

```
public int hashCode() {
  return this.x + this.y;
}
```

Yes

4.)

```
public int hashCode() {
  return count;
}
```

No. Equal objects don’t return the same hash code

5.)

```
public int hashCode() {
  return 4;
}
```

Valid but bad

This is a hashcode for a binary string which only contains characters that are ‘0’ and ‘1’. Is this a good hash code and why? You may have seen something similar from lecture.

```
public class BinString {
      public int hashCode() {
            int total = 0;
            String s = this.str;
            for (int i = 0; i < s.length; i++) {
                  char c = s.charAt(i);
                  total = 2 * total + (c - '0');
            }
            return total;
      }
}
```

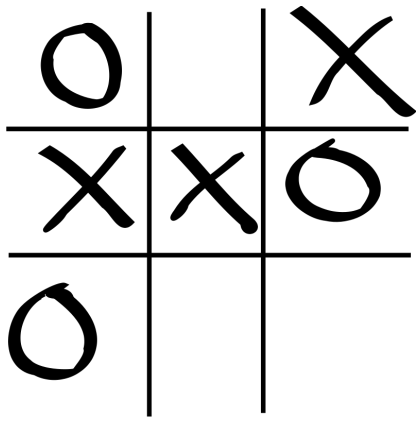
A ternary string consists of only of ‘0’,’1’,’2’ instead of just ‘0’ and ‘1’. Implement a hash code for TriString that encodes the TriString as a int.

```
public class TriString {
      String str;

      public TriString(String triStr) {
            str = triStr;
      }

      public int hashCode() {
            int total = 0;
            String s = this.str;
            for (int i = 0; i < s.length; i++) {
                  char c = s.charAt(i);
                  total = 3*total + (c - '0');
            }
            return total;
      }
}
```

The class TicTacToe is a game state of the game tic tac toe, which has a 3x3 grid where every square is either empty or contains an ‘X’ or an ‘O’. Empty squares are represented as ‘0’, ‘O’s are represented as ‘1’, and ‘X’s are represented as ‘2’.



```
public class TicTacToe {
	char[][] board;

	public TicTacToe() {
		board = new char[3][3];
	}

	public TriString toTriString() {
		String triString = “”;
		for (int i = 0; i < board.length; i++) {
			for (int j = 0; j < board[0].length; j++) {
				triString += board[i][j];
			}
		}
		return new TriString(triString);
	}

	public int hashCode() {
		return this.toTriString().hashCode();
	}
}
```

Extra Question!

Design a hashcode for strings. Assume the string is only made up of lower case characters and spaces.

```
public class String {
      public int hashCode() {
            int total = 0;
            String s = this;
            for (int i = 0; i < s.length(); i++) {
                  char c = s.charAt(i);
                  total = 31*total + (c - 'a');
            }
            return total;
      }
}
```

Number must be a prime because it's better for hashing (abstract this away by mentioning 170)