CS 314 Final Review — Hash Table Resize — **Solution**

Hash Tables

Write a private instance method `resize` for the provided `HashTable314` class which will double the size of the internal array and re-index the elements in the hash table. This hash table implementation uses linked bucket nodes to handle hash collisions. These linked lists are **not** kept in sorted order. Elements in the hash table are placed in the bucket corresponding to their hash code modulo the length of the array.

Your `resize` method should also return the magnitude (absolute value) of the largest change in index as a result of the resize. For example, if an element was originally at index 1 in `con` but after a call to `resize` its index became 11, that is a change in index of 10. Your method will return the largest of these changes.

This problem is about Hash Tables which use buckets for collisions. If you would like practice with a Hash Table which uses linear probing, look at the last problem on the Spring 2019 final.

```
/* Pre: none
 * Post: con will be twice as large, size is unchanged
 */
 private int resize();
```

You may use the following `HashTable314` implementation.

```
public class HashTable314<E> {
    private static final int INITIAL_CAPACITY = 10;

    private BucketNode<E>[] con;
    private int size;

    private static class BucketNode<E> {
      private E data;
      private BucketNode<E> next;
    }
}
```

You may use the absolute value method (`abs`) from the `Math` class.
You may not use or assume any other methods exist in the `HashTable314` class.
Do not use any other Java classes or methods.

```
 /* Pre: none
  * Post: con will be twice as large, size is unchanged
  */
  private int resize(){
    BucketNode<E>[] temp = (BucketNode<E>[]) new Object[con.length * 2];
    int largestChange = 0;
    for(int i=0; i < con.length; i++){
        while(con[i] != null){
            BucketNode<E> n = con[i];
            con[i] = n.next;
            int index = Math.abs(n.data.hashCode() % temp.length);
            n.next = temp[index];
            temp[index] = n;

            //Find largest change
            int change = Math.abs(i - index);
            largestChange = Math.max(largestChange, change);
        }
    }
    return largestChange;
}
```