Problem 11-3

The class `MyHashtable` described in class requires one more refinement: The put method must not continue inserting elements into the hashtable if the hashtable is too full.

The "fullness" of the hashtable is measured by its *load factor*. The load factor is the total number of entries in the hashtable divided by the table size. In MyHashtable, a default maximum load factor is set at the value 5. This means that the number of entries in the hashtable must never exceed 5 times the table size.

This requirement is not being enforced in the version of MyHashtable that is in your folder. In this exercise, you will enforce this requirement.

You will need to do two things: First, create an additional method `rehash()`. Second, rewrite the `put` method so that it calls `rehash()` at the appropriate time.

The signature of the new method that you will add is

```
private void rehash()
```

This `rehash` method does the following:

1. It creates a new temp array `tableTemp`, having size twice the current size of `table,` and updates the variable `tableSize` with these new sizes.
2. It examines each key stored in `table` and populates the new table in the following way: It performs `hashCode` and `hash` on each key – producing in each case a new array index `i` – and then it inserts the `Entry` into the linked list in slot `i` of `tableTemp`.
3. Finally, it replaces the old `table` with this temp table.

The other thing to do is to modify the `put` operation so that if the loadFactor of the table is greater than or equal to 5, it calls `rehash` before executing its usual steps of operation.