7. [3] I grabbed some code from the Internet for my linear probing based hash table at work because the Internet's always right (totally!). The hash table works, but once I put more than a few thousand entries, the whole thing starts to slow down. Searches, inserts, and contains calls start taking *much* longer than O(1) time and my boss is pissed because it's slowing down the whole application services backend I'm in charge of. I think the bug is in my rehash code, but I'm not sure where. Any ideas why my hash table starts to suck as it grows bigger?

Should just send in a reference to the array → thats bad ....

copy the whole array 0-n

fast

```
1  /**
2  * Rehashing for linear probing hash table.
3  */
4  void rehash( )
5  {
6      ArrayList<HashItem<T>> oldArray = array;
7
8      array = new ArrayList<HashItem<T>>( 2 * oldArray.size() );
9
10     for( int i = 0; i < array.size(); i++ )
11         array.get(i).info = EMPTY;
12 // Copy old table over to new larger array
13     for( int i = 0; i < oldArray.size(); i++ ) {
14         if( oldArray.get(i).info == FULL )
15         {
16             addElement(oldArray.get(i).getKey(),
17                     oldArray.get(i).getValue());
18         }
19     }
20 }
```

(Line 6)

Because of ArrayList<hashItem<T>> old array

This line is deep copying the whole oldarray every time it has to rehash causing increased items as the hash table increases in size