Exercise 3.4.15: How many compares could it take, in the worst case, to insert N keys into an initially empty table, using linear probing with array resizing?

Background Info:
Recall Linear probing tries to avoid collisions by placing the value to another position while Seperate Chaining allows the collisions to happen and ends up putting the collided values in the same index in the form of a linked list. This results in search for certain items in a particular index be dependent solely on the size of the LinkedList since it has to check each index. Thus $O(N)$ worst case for search for seperate chaining where N is size of the longest linkedlist at a index.

Linear probing allocates large memory to be able to avoid collisions. This is only good if we have a large number of elements. Probing and hashing are deterministic. Seperate chaining as explained above is useful if not trying to waste extra memory. Linear probing works like this:

- you find a spot that is occupied by a value, you move to the next spot and see if you can place your value there.
- If not, you keep continue moving which can result in checking for (n-1) entries.

Length of cluser for search of Linear Probing requires us to check for **worst case** element $(n-1)$ as we have to search from beginning to end for all n-elements.

However, if the ideal case of uniform assumption occurs, then we can find a space to place our value in the hashtable with average case $O(1)$ for insert and search. **Uniform hashing assumption** is when we have equal probability of placing an element/item in each space as $\frac{N}{M}$ where $M > N$. M is the size of the hashtable while N is the number of elements we have. If $M > N$, then the $\frac{N}{M}$ gives $\sim O(1)$.

Solution:

The problem does not mention initial size of the hashtable or the load value for which it wants us to double the hashtable size.

We can assume that the hashtable will double when half full.

As we insert values all the way up till $(\frac{N}{2})_{cn}$ element, the number of compares will increase by 1 each time. ie; 1 for first insert, 2 for second insert, ... until $\frac{N}{2}$

In the worst-case, each element will probe all previously inserted elements; even if this happens only for the N=4 last elements before the next resizing (the previous one happened when the total number of elements was N=4), then the number of compares is equivalent to $O(N^2)$ complexity.