

### 3. Double Hashing

$f(i) = i * \text{hash}_2(x);$  /\*  $\text{hash}_2(x)$  is the 2<sup>nd</sup> hash function \*/

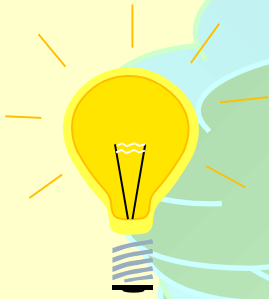
❶  $\text{hash}_2(x) \neq 0$  ; ❶ make sure that all cells can be probed.

👉 **Tip:**  $\text{hash}_2(x) = R - (x \% R)$  with  $R$  a prime smaller than TableSize, will work well.

**Note:** ❶ If double hashing is correctly implemented, simulations imply that the **expected** number of probes is almost the same as for a **random** collision resolution strategy.

❷ Quadratic probing does not require the use of a second hash function and is thus likely to be **simpler and faster** in practice.

## §5 Rehashing



- ☞ Build another table that is about twice as big;
- ☞ Scan down the entire original hash table for non-deleted elements; Then what can we do? ag...
- ☞ Use a new function to hash those elements into the new table.

If there are  $N$  keys in the table, then  $T(N) = O(N)$

**Question: When to rehash?**

**Answer:**

- ① As soon as the table is half full
- ② When an insertion fails
- ③ When the table reaches a certain load factor

**Note:** Usually there should have been  $N/2$  insertions before rehash, so  $O(N)$  rehash only adds a constant cost to each insertion.

However, in an interactive system, the unfortunate user whose insertion caused a rehash could see a slowdown.

**Read Figures 7.23  
for detailed implementation of rehashing.**