Hashing

**Application:** Maintain a set of items and support the three operations of **INSERT**, **MEMBER**, and **DELETE**.

**Idea:** Use a *hash function* \( h \) to map a data item \( d \) into a hash table array \( A[0]...A[m−1] \). Each position is called a *bucket*, and has a pointer to a linked list of all items that have hashed to that position.
Notation:

$A = \text{The hash table};$ each location of the array $A[0] \ldots A[m–1]$ (called a "bucket") contains a pointer to a linked list of all items stored there.

$h = \text{A function that maps a data item } d \text{ to an integer } 0 \leq h(d) < m.$

Basic operations:

**MEMBER($d$):** Search the bucket at $A[h(d)]$.

**INSERT($d$):** Do MEMBER($d$) and if $d$ is not present, add $d$ to the bucket at $A[h(d)]$.

**DELETE($d$):** Do MEMBER($d$) and if $d$ is present, remove $d$ from the bucket at $A[h(d)]$. 
The MOD hash function:

\[ a \text{ MOD } b \] denotes the remainder when \( a \) is divided by \( b \):

\[
(a \text{ MOD } b) = a - \left\lfloor \frac{a}{b} \right\rfloor b
\]

It is always an integer in the range 0 to \( b-1 \) when \( a \) and \( b \) are positive integers.

When \( d \) is an integer, the standard MOD hash function computes:

\[ h(d) = d \text{ MOD } m \]

To avoid patterns when \( d \) and \( m \) have factors in common, \( m \) should be a prime number. Alternately, for a large prime \( q>m \), compute:

\[ h(d) = (d \text{ MOD } q) \text{ MOD } m \]