HASHING CONCEPTS

Hashing needs 2 algorithms:

1. A **hash function** (HF) determines the record's homeAddress, based on the record's key field and the overall MAX_N_LOC supplied to the HF.
2. A **collision resolution algorithm** determines where collisions are stored and how they'll be accessed.

All records which hash to the same homeAddress are called a "synonym family".
The first record hashing to a particular homeAddress is stored in that homeAddress location.
A record is a collision if it can NOT be stored in its homeAddress, because another record is already there.
Collisions may arise during the initial loading in Setup or during transaction processing in UserApp.

Static hashing:

- Static hashing (*vs. dynamic hashing*) means that the storage size of the table or file is already determined at compile time (*rather than at run-time, and thus would be increased, as needed*)
- "Storage size" refers to the number of storage locations:
    - 0 through MAX_N_LOC-1 for internal storage (i.e., subscripts in an array)
    - 1 through MAX_N_LOC for external storage (i.e., RRNs in a relative file)
- This size usually refers to:
    - the entire table/file for "EMBEDDED overflow" (collisions)
    - the home area for "SEPARATE overflow" (collisions)
- MAX_N_LOC is a named constant, since it is something that is often changed during project development or maintenance once performance is evaluated. DataTable class methods, whether used by Setup or UserApp MUST use the same exact MAX_N_LOC.

Hash Functions should always be :

- named hashFunction since their code often needs to be tweaked to enhance performance during development or maintenance
- a single, callable method – so there is only ONE COPY of the code anywhere in the project, making for consistency of HF and ease of changing the code
- be entirely contained within a method body, so it can easily be changed (and not just written as in-line code because it's only a couple lines of code)

REMINDER ON **STARTING & ENDING LOCATIONS** (for static hashing)

- 0 through MAX_N_LOC-1 for internal storage (i.e., subscripts in an array)
- 1 through MAX_N_LOC for external storage (i.e., RRNs in a relative file)

CRAs include 2 separate design decisions (for static hashing)

1) Embedded (in home area) OR Separate (in a collision area, separate from home area)
2) CHOOSE 1: Linear or ReHash or Chaining

A separate collision area may be in:

- a physically separate file/table
- OR n the same file/table as the home area, but after the end of the homeArea, starting in location:
    - MAX_N_LOC (for internal storage)
    - MAX_N_LOC + 1 (for external storage)

Separate collision areas allow for a theoretically infinite number of storage locations.
Embedded collision area limit the number of possible inserts.
Chaining with separate collision area is generally the most time-efficient option, by far.
On the average, assuming a reasonably good hash function, it results in
O(1) for a single query (the time component) with actual time being ~1.5
and 95% packing density (the space component).