**An Efficient Imperative Map**

Hashtable

* compute hash of keys, determiines the bucket
* storing key-value pairs in buckets
  + buckets as linked lists, or
  + buckets within the array, with a probing function that fills array
* for constant time we have large enough array, collisions rare
* as table grows, we resize array and rehash everything

Undo stack needed for push/pop:

* push a marker on stack when we enter scope (procedure, block)
* all entries added to stack
* exiting scope:
  + pop until the last mark
  + delete each entry that is popped (or restore previous one)

**References**

* [Course by Leiserson from MIT OpenCourseware](http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/LectureNotes/index.htm)