Hashing alternatives

- chaining requires dynamic memory, not always available
  - open addressing/linear probing
  - double hashing

- linear probing: are there problems?

  ```
  int loc = Hash(key) % TABLE_SIZE;
  int newVal = 1;
  while (isEmpty(loc))
  {
    loc += newVal; if (loc >= TABLE_SIZE) loc = 0;
  }
  table[loc] = key;
  ```

  Hash(key) --> key[0]

  ant, ape, bear, bat, bee, cat
more linear probing, alternatives

- problems with linear probing:
  - deletions
  - clustering (parking spaces analogy?)

- quadratic probing/ double hashing
  ```c
  int loc = Hash(key) % TABLE_SIZE;
  int newVal = _______
  while (isEmpty(loc))
  {
      loc += newVal;
      if (loc >= TABLE_SIZE) loc = 0;
  }
  table[loc] = key;
  ```

- newval = $1^2$, $2^2$, $3^2$, ...
- newval = Hash2(key) ;
Best Hashing method, issues

- chaining is easy to implement
  - search/delete straightforward linked-list operations
  - table has room to grow (to a degree)
  - compare with linear/quadratic probing, double hashing

- what can we do if/when table is full?
  - rehash to a larger table
  - advanced methods (e.g., extendible hashing)

- implementation issues
  - access to values in table
  - member functions, storage of keys and values
Implementation issues

- **hash-table is mapping of key --> value, e.g., word/# occurrences**
  - includesKey(), getKey(), get(key,value), ...

- **access to hash-table values**
  - hash-table applies a function to every key/value
    - print function
    - change values int able
    - what about summing values?
  - hash-table applies an object to every key/value
    - objects have internal state
  - supply each key/value one-at-a-time to client program