- get(k):
  $i \leftarrow h(k)$
  while $A[i] \neq$ NULL **do**
    if $A[i]$.key $= k$ **then**
      return $A[i]$
    $i \leftarrow (i+1) \bmod N$
  return NULL
- put(k, v):
  $i \leftarrow h(k)$
  while $A[i] \neq$ NULL **do**
    if $A[i]$.key $= k$ **then**
      $A[i] \leftarrow (k, v)$ // replace the old (k, v')
    $i \leftarrow (i+1) \bmod N$
  $A[i] \leftarrow (k, v)$
- remove(k):
  $i \leftarrow h(k)$
  while $A[i] \neq$ NULL **do**
    if $A[i]$.key $= k$ **then**
      $temp \leftarrow A[i]$
      $A[i] \leftarrow$ NULL
      Call Shift(i) to restore A to a stable state without k
      return $temp$
    $i \leftarrow (i+1) \bmod N$
  return NULL
- Shift(i):
  $s \leftarrow 1$ // the current shift amount
  while $A[(i+s) \bmod N] \neq$ NULL **do**
    $j \leftarrow h(A[(i+s) \bmod N]$.key) // preferred index for this item
    if $j \notin (i, i+s] \bmod N$ **then**
      $A[i] \leftarrow A[(i+s) \bmod N]$ // fill in the "hole"
      $A[(i+s) \bmod N] \leftarrow$ NULL // move the "hole"
      $i \leftarrow (i+s) \bmod N$
      $s \leftarrow 1$
    else
      $s \leftarrow s+1$

One alternative to the shifting done above for remove(k) is to replace the deleted item by a special "deactivated item" object. With this special marker possibly occupying buckets in our hash table, we would then need to modify our search algorithm for remove(k) or get(k), so that the search for a key k should skip over deactivated items and continue probing until reaching the desired item or an empty bucket. Likewise, the put(k, v) algorithm should stop at a deactivated item and replace it with the new item to be inserted. (See Exercise C-6.1.)