7. Write a C++ function to delete the pair with key k from a hash table that uses linear probing. Show that simply setting the slot previously occupied by the deleted pair to empty does not solve the problem. How must *Get* (Program 8.4) be modified so that a correct search is made in the situation when deletions are permitted? Where can a new key be inserted?

```
template <class K, class E>
pair<K, E>* LinearProbing<K, E>::Get(const K& k)
{// Search the linear probing hash table ht (each bucket has exactly one slot) for k.
// If a pair with this key is found, return a pointer to this pair; otherwise, return 0.
   int i = h(k);  // home bucket
   int j;
   for (j = i; ht[j] && ht[j]->first != k;) {
      j = (j + 1) % b;          // treat the table as circular
      if (j == i) return 0;     // back to start point
   }
   if (ht[j]->first == k) return ht[j];
   return 0;
}
```

Program 8.4: Linear probing

When the "for" function runs to the slot previously occupied by the deleted pair, it will be stopped and return value 0, but it probably exists in the slot later.

```
#define Num 10000

pair<K,E>* LinearProbing<K,E>::Get(const K& k){
    int i=h(k);
    for(int j=i; ;){
        if(!ht[j])
            j=(j+1)%b;
        else if(ht[j]->first==k)
            return(ht[j];
        else
            j=(j+1)%b;
        if(i==j) return 0;
    }
```

if there is a ht[j] not existing, insert in this position.
if after the cycle run and not find a position empty, insert in ht[b].