0	
1	
2	
3	

Example:
Start buckets array with size 4, containing null
ASCII code as hash function ("a" = 97)

set("b", 70) # note 98 % 4 is 2
set("f", 90)
set("f", 100)

How many elements in bucket 1?
A: 0   B: 1   C: 2   D: 3   E: more than 3

How many elements in bucket 2?
A: 0   B: 1   C: 2   D: 3   E: more than 3

How many elements in bucket 3?
A: 0   B: 1   C: 2   D: 3   E: more than 3

How many entries are checked when doing set("f", 100)?
A: 0   B: 1   C: 2   D: 3   E: more than 3

What will the result of get("f") be after this sequence?
A: 70   B: 90   C: 100   D: null   E: an error

Example continued...

set("c", 40)

Which bucket is "c" stored in?

A: 0   B: 1   C: 2   D: 3   E: it causes an error

A HashTable<Key, Value> using Linear Probing has:
- size: an int
- buckets: an array of Entries (not of lists of Entries!)
- hash: a hash function for the Key type

An Entry is a single {key: value} pair.

```
void set(key, value):
  if loadFactor > 0.67: expandCapacity()
  hashed = hash(key)
  index = hashed % array length
  while this.buckets[index] != null:
    b = this.buckets[index]
    if b.key.equals(key):
      b.value = value
      return
    index += 1


  // key not in table, add it at first index containing null
  this.buckets[index] = {key: value}




Value get(key):
  hashed = hash(key)
  index = hashed % this.buckets.length
  while this.buckets[index] != null:
    b = this.buckets[index]
    if b.key.equals(key): return b.value
    index += 1


  // haven't found the key
  return null/throw exception


void expandCapacity():
  newEntries = new Entry[this.buckets.length * 2];
  oldEntries = this.buckets
  this.buckets = newEntries
  this.size = 0
  for each entry {k:v} in oldEntries:
    this.set(k, v)
```

```
public class AList<E> implements List<E> {

  E[] elements;
  int size;

  @SuppressWarnings("unchecked")
  public AList() {
    this.elements = (E[])(new Object[2]);
    this.size = 0;
  }

  public void add(E s) {
    expandCapacity();
    this.elements[this.size] = s;
    this.size += 1;
  }

  @SuppressWarnings("unchecked")
  private void expandCapacity() {
    int currentCapacity = this.elements.length;
    if(this.size < currentCapacity) { return; }

    E[] expanded = (E[])(new Object[currentCapacity * 2]);

    for(int i = 0; i < this.size; i += 1) {
      expanded[i] = this.elements[i];
    }
    this.elements = expanded;
  }
}
```

If we add 6 elements to an empty AList, what is the **sum of all the lengths of arrays created in (including constructor and expandCapacity)?**

A: 8   B: 10   C: 12   D: 14   E: 16

If we add 6 elements to an empty AList, what is the **total number of times an element is copied in expandCapacity?**

A: 6   B: 8   C: 10   D: 12   E: 16

If we add 20 elements to an empty AList, **how many times is expandCapacity called?**

A: 2   B: 3   C: 4   D: 5   E: 6

If we add 20 elements to an empty AList, **what is the length of the array created in each of those calls to expandCapacity?** (open-ended, no multiple-choice)