Question 1: Assuming open addressing, what is the running time of inserting an element into a hash table with *n* elements and a load factor of 0.5?

Question 2: Assuming open addressing, what is the running time of inserting an element into a hash table with *n* elements and only 10 empty slots?

Question 3: Assuming separate chaining, what is the running time of inserting an element into a hash table with *n* elements and a load factor of *f*?

Question 4: What is the running time of the following code:

```
k = 0;
for (i = 0; i < 1000*n; i++) k++;
```

Question 5: What is the running time of the following code:

```
k = 0;
for (i = 0; i < 2*n; i++) {
  for (j = 0; j < 10000; j++) k++;
}
```

Question 6: What is the running time of the following code:

```
k = 0;
for (i = 1; i < (n * n); i *= 2) k++;
```

Answers to clicker questions

Question 1: When half of the table is empty, your expected number of probes to find an empty slot is 2, so the answer is *O(1)*.

Question 2: With only 10 empty slots, your expected number of probes becomes *n/10*, so the answer is *O(n)*. I won't tell you how to calculate this, just trust me.

Question 3: With a load factor of *f*, the average size of the vector in a hash table entry is *f*, so the answer is *O(f)*.

Question 4: The loop runs *1000n* times, which is *O(n)*.

Question 5: The outer loop runs *2n* times, so its number of iterations is *O(n)*. The inner loop runs 10,000 times so its number of iterations is *O(1)*. *O(n) * O(1) = O(n)*.

Question 6: The loop runs $log(n^2)$ times.

$$log(n^2) = 2*log(n)$$

Therefore, the loop's running time is *O(log n)*.