CS62: Fall 2025 | Lecture #19 (Hashtables Pt 2) worksheet | Prof. Li

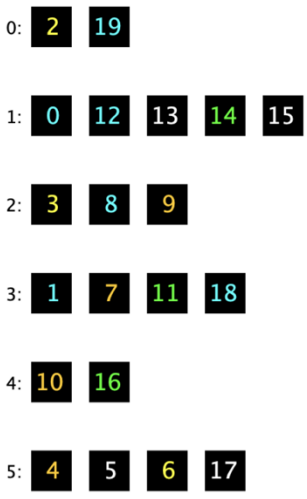
1. Fill in the blanks to implement get() in a separate chaining hash table. You can assume you have access to the `hash()` method, and an instance variable called `table` which is an array of Nodes, where Nodes contain a key, value, and next pointer (they are Nodes in a SLL).

```
public Value get(Key key) {

   int i = _____________________; //hash the key

   for (_____________________________________) { //go through linked list

      if (_____________________) { //if the keys match

          return _________________; //return the value

      }

   }

   return null;

}
```

2. Insert the key-value pairs (47, 0), (3, 1), (28, 2), (14, 3), (9,4), (47,5) into an open addressing hash table of size m = 7.
   Assume the hash function is calculated as key % m.

3. Assuming m=9, insert keys 3, 9, 18, 0, 4, 36 in a quadratic probing hash table. Assume h(k) = key % m and h(k, i) = (h(k) + i^2) % m. What is the load factor?

4. Suppose we have a separate chaining hash table of ColoredNumbers. The hashCode is the memory location while equals() is overridden to check if the ColoredNumber's num attributes are equal.




```
ColoredNumber zero = new ColoredNumber(0);
hs.add(zero);
```

What can happen when we call hs.add(zero)?

A. We add another 0 to bin zero.
B. We add another 0 to bin one.
C. We add another 0 to some other bin.
D. We do not get a duplicate zero.