

CS62: Spring 2025 | Lecture #21 (Hashtables Pt 2) worksheet | Jingyi Li

1. 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 $\text{key} \% m$.

2. 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.

0: 2 19

1: 0 12 13 14 15

2: 3 8 9

3: 1 7 11 18

4: 10 16

5: 4 5 6 17

```
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.

3. 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;  
}
```