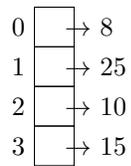# 1 External Chaining

Consider the following External Chaining Hash Set below, which doubles in size when the load factor reaches 1.5. Assume that we're using the default hashCode for integers, which simply returns the integer itself.

```
0 |    |→ 8
1 |    |→ 25
2 |    |→ 10
3 |    |→ 15
```

(a) Draw the External Chaining Hash Set that results if we insert 18.

(b) Draw the External Chaining Hash Set that results if we insert 5 after the insertion done in part (a).

2 Invalid Hashes

For both parts below, suppose we are trying to hash the following class:

```java
import java.util.Random;
class Point {
    private int x;
    private int y;
    private static count = 0;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
        count += 1;
    }
}
```

(a) Which of the hashCodes are invalid?

   (i)
```java
public void hashCode() {
    System.out.print(this.x + this.y);
}
```

  (ii)
```java
public int hashCode() {
    Random randomGenerator = new Random();
    return randomGenerator.nextInt(Int);
}
```

 (iii)
```java
public int hashCode() {
    return this.x + this.y;
}
```

 (iv)
```java
public int hashCode() {
    return 4;
}
```

  (v)
```java
public int hashCode() {
    return count;
}
```

(b) *Extra:* Suppose we know all the `Points` have x and y coordinates between 0 and 10, inclusive. Suggest a **good** hashCode method.

3 Hashing Gone Crazy

For this question, use the following TA class for reference.

```java
public class TA {
    int charisma;
    String name;
    TA(String name, int charisma) {
        this.name = name;
        this.charisma = charisma;
    }
    @Override
    public boolean equals(Object o) {
        TA other = (TA) o;
        return other.name.charAt(0) == this.name.charAt(0);
    }
    @Override
    public int hashCode() {
        return charisma;
    }
}
```

Assume that the `hashCode` of a `TA` object returns `charisma`, and the `equals` method returns `true` if and only if two `TA` objects have the same first letter in their `name`.

Assume that the `ECHashMap` is a `HashMap` implemented with external chaining as depicted in lecture. The `ECHashMap` instance begins at size 4 and, for simplicity, does not resize. Draw the contents of `map` after the executing the insertions below:

```java
ECHashMap<TA, Integer> map = new ECHashMap<>();
TA sohum = new TA("Sohum", 10);
TA vivant = new TA("Vivant", 20);
map.put(sohum, 1);
map.put(vivant, 2);

vivant.charisma += 2;
map.put(vivant, 3);

sohum.name = "Vohum";
map.put(vivant, 4);

sohum.charisma += 2;
map.put(sohum, 5);

sohum.name = "Sohum";
TA shubha = new TA("Shubha", 24);
map.put(shubha, 6);
```