```
private static <K, V> void moveToFront(Queue<Pair<K, V>> q, K key) {

Pair<K,V> temp = null;

for (int I =0; I < q.length(); i++) {

                        temp = q.dequeue();

                        if (temp.key().equals(key)) {

                                    I = q.length();

                        }

                        q.enqueue(temp);

            }

}
```

```
    @Test
    public void addTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2");
        Map<String, String> ref = this.createFromArgsRef("1", "1", "2", "2",
                "3", "3");

        test.add("3", "3");

        assertEquals(ref, test);
    }

    @Test
    public void removeTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2",
                "3", "3");
        Map<String, String> ref = this.createFromArgsRef("1", "1", "2", "2");

        test.remove("3");

        assertEquals(ref, test);
    }

    @Test
    public void removeAnyTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2",
                "3", "3");
        Map<String, String> ref = this.createFromArgsRef("2", "2", "3", "3");

        test.removeAny();
        assertEquals(ref, test);
    }

    @Test
    public void valueTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2",
                "3", "3");

        assertEquals("2", test.value("2"));
    }

    @Test
    public void hasKeyTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2",
                "3", "3");

        assertEquals(true, test.hasKey("1"));
        assertEquals(true, test.hasKey("2"));
        assertEquals(true, test.hasKey("3"));
        assertEquals(false, test.hasKey("4"));
        assertEquals(false, test.hasKey("0"));
    }

    @Test
    public void sizeTest() {

        Map<String, String> test = this.createFromArgsTest("1", "1", "2",
"2",
                "3", "3");

        assertEquals(3, test.size());
    }
```

Statement	Variable Values
`Map<String, Integer> m = new Map1L<>();`	
	m = `<>`
`m.add("one", 1);`	
	m = `{("one", 1)}`
`m.add("zero", 0);`	
	m = `{("one", 1), ("zero", 0)}`
`m.add("negative one", -1);`	
	m = `{("one", 1), ("zero", 0), ("negative one", -1)}`
`Pair<String, Integer> p = m.remove("zero");`	
	m = `{("one", 1), ("negative one", -1)}` p = `("zero", 0)`
`m.remove("one");`	
	m = `{("negative one", -1)}` p = `("zero", 0)`
`m.add("cipher", p.value());`	
	m = `{("negative one", -1), ("cipher", 0)}` p = `("zero", 0)`
`m.add(p.key(), p.value());`	
	m = `{("negative one", -1), ("cipher", 0), ("zero", 0)}` p = `("zero", 0)`
`m.remove("negative one");`	
	m = `{("cipher", 0), ("zero", 0)}` p = `("zero", 0)`
`m.remove("cipher");`	
	m = `{("zero", 0)}` p = `("zero", 0)`
`p = m.removeAny();`	
	m = `{}` p = `("zero", 0)`