private static <K, V> void bringToFront(Queue<Pair<K, V>> queue, K targetKey) {

assert queue != null : "Violation of: queue is not null";

assert targetKey != null : "Violation of: targetKey is not null";

Queue<Pair<K, V>> tempQueue = queue.newInstance();

while (queue.length() != 0 && !queue.front().key().equals(targetKey)) {

tempQueue.enqueue(queue.dequeue());

}

queue.append(tempQueue);

}

/\*\*

\* Test constructor.

\*/

@Test

public final void verifyConstructor() {

Map<String, String> actualMap = this.constructorTest();

Map<String, String> expectedMap = this.constructorRef();

assertEquals(actualMap, expectedMap);

}

/\*\*

\* Test adding to a non-empty map.

\*/

@Test

public final void verifyAddToNonEmptyMap() {

Map<String, String> actualMap = this.createFromArgsTest("apple", "fruit", "carrot", "vegetable");

Map<String, String> expectedMap = this.createFromArgsRef("apple", "fruit", "carrot", "vegetable", "banana", "fruit");

actualMap.add("banana", "fruit");

boolean isConsistent = false;

for (Pair<String, String> pair : expectedMap) {

if (actualMap.hasKey(pair.key()) && actualMap.hasValue(pair.value()) && actualMap.key(pair.value()).equals(pair.key())) {

isConsistent = true;

}

}

assertTrue(isConsistent);

}

/\*\*

\* Test removing an entry from the map.

\*/

@Test

public final void verifyRemoveEntry() {

Map<String, String> actualMap = this.createFromArgsTest("dog", "mammal", "sparrow", "bird");

Map<String, String> expectedMap = this.createFromArgsRef("sparrow", "bird");

actualMap.remove("dog");

assertTrue(!actualMap.hasKey("dog") && actualMap.equals(expectedMap));

}

/\*\*

\* Test removing any entry from the map.

\*/

@Test

public final void verifyRemoveAnyEntry() {

Map<String, String> actualMap = this.createFromArgsTest("cat", "mammal", "dolphin", "mammal", "eagle", "bird");

int initialSize = actualMap.size();

actualMap.removeAny();

int newSize = actualMap.size();

assertTrue(newSize == initialSize - 1);

}

/\*\*

\* Test retrieving a value by key.

\*/

@Test

public final void verifyGetValue() {

Map<String, String> actualMap = this.createFromArgsTest("rose", "flower", "oak", "tree");

String value1 = actualMap.value("rose");

String value2 = actualMap.value("oak");

assertTrue(value1.equals("flower") && value2.equals("tree"));

}

/\*\*

\* Test checking if keys exist in the map.

\*/

@Test

public final void verifyHasKey() {

Map<String, String> actualMap = this.createFromArgsTest("lion", "animal", "tulip", "flower");

assertTrue(actualMap.hasKey("lion") && actualMap.hasKey("tulip") && !actualMap.hasKey("animal"));

}

/\*\*

\* Test the size of the map.

\*/

@Test

public final void verifySize() {

Map<String, String> actualMap = this.createFromArgsTest("apple", "fruit", "carrot", "vegetable", "banana", "fruit");

int actualSize = actualMap.size();

int expectedSize = 3;

assertEquals(actualSize, expectedSize);

}