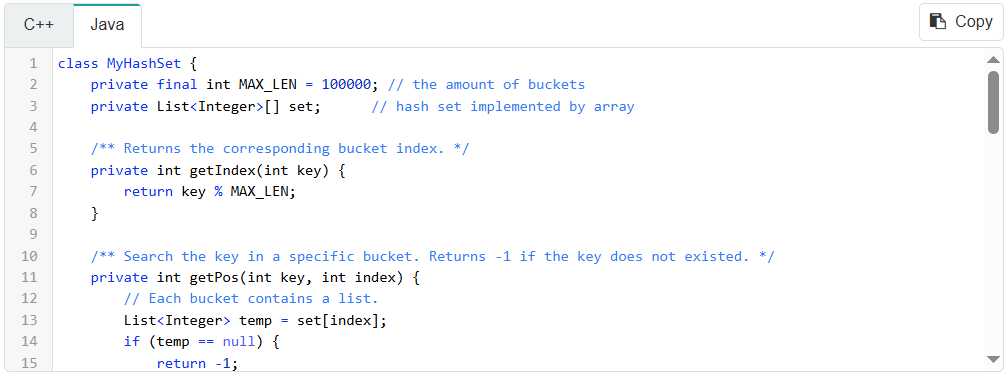
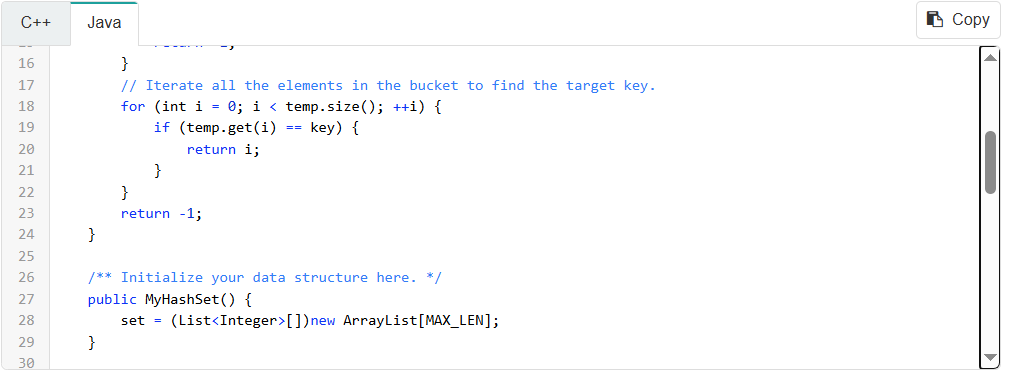
**Design Hash Table - Solution**

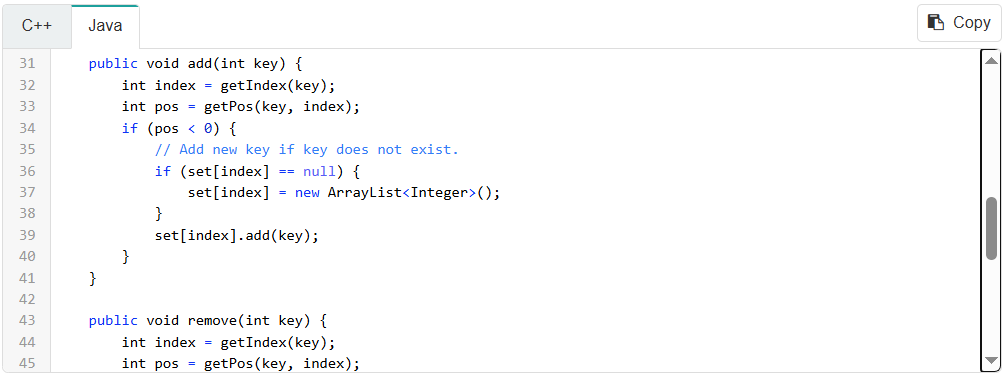
[Report Issue](https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

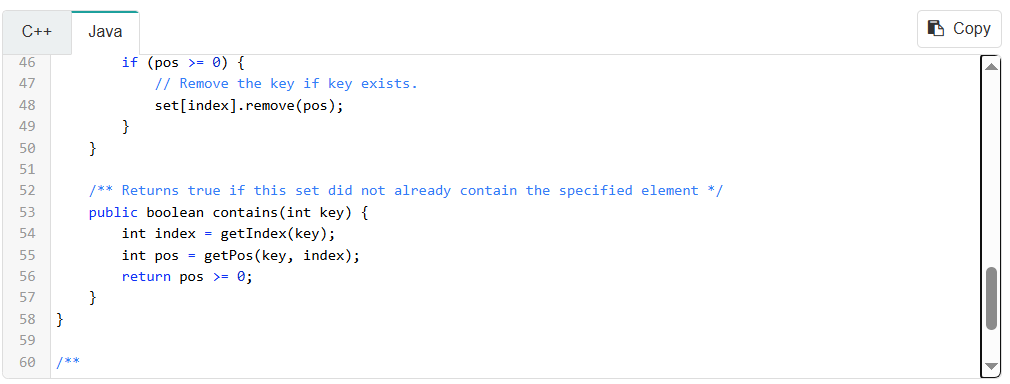
Here are C++ and Java solutions for your reference. In our solution, we use an array to represent the hash set. Each element in the array is a bucket. And in each bucket, we use the array list (or vector in C++) to store all the values.

*Hash Set*

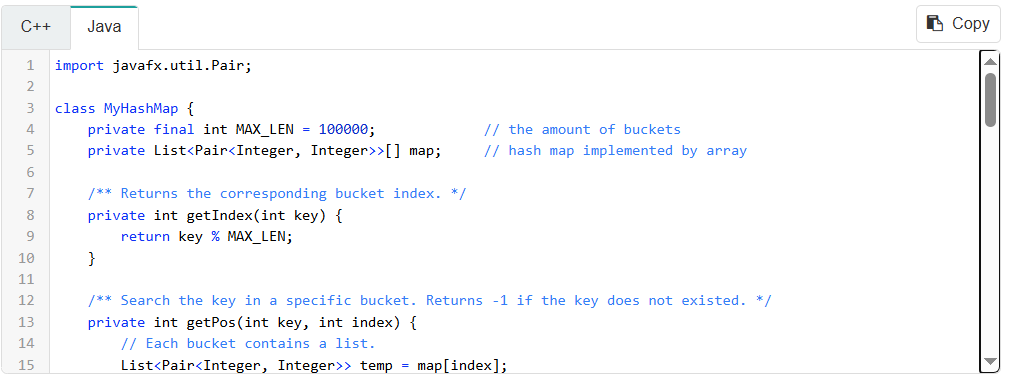


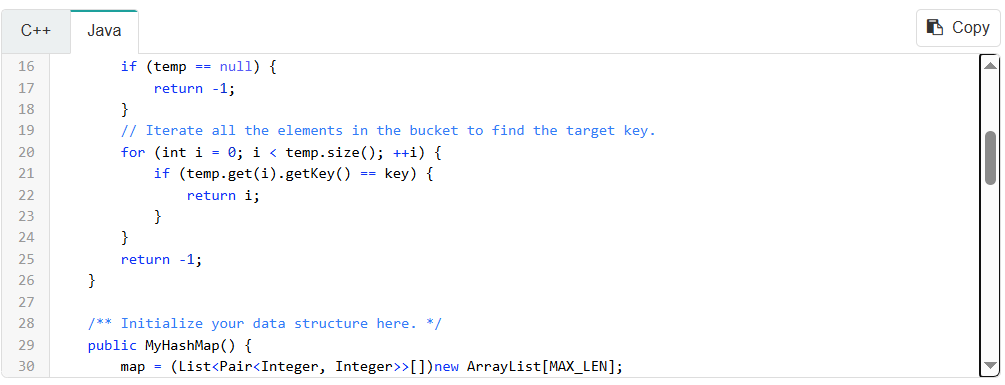


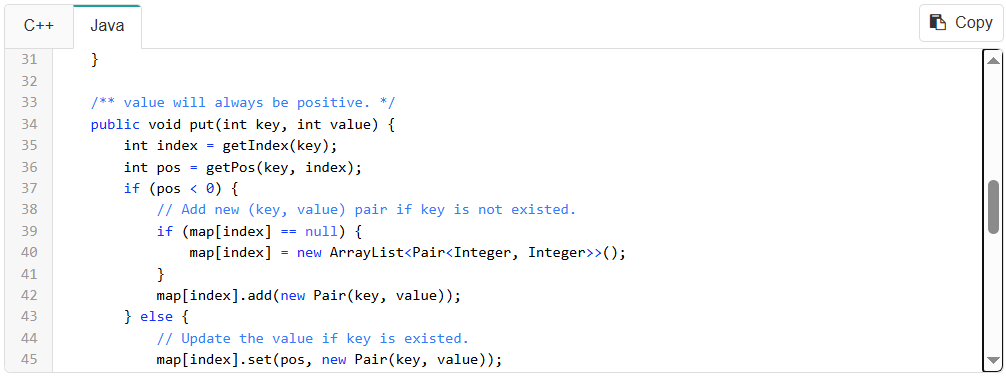


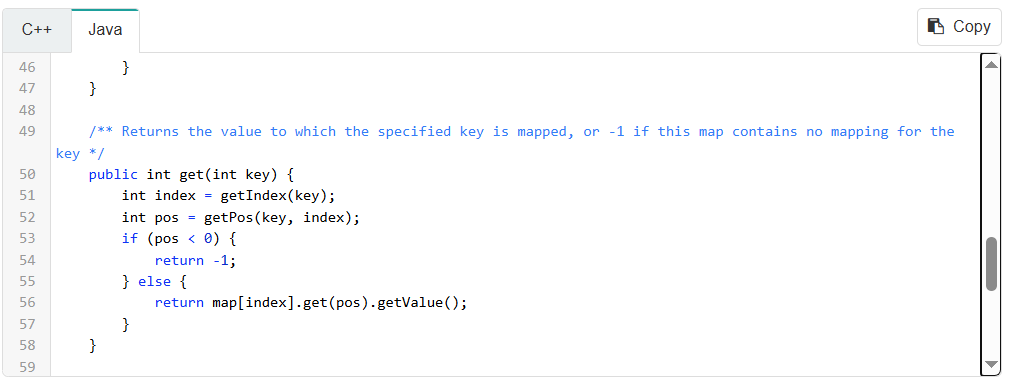


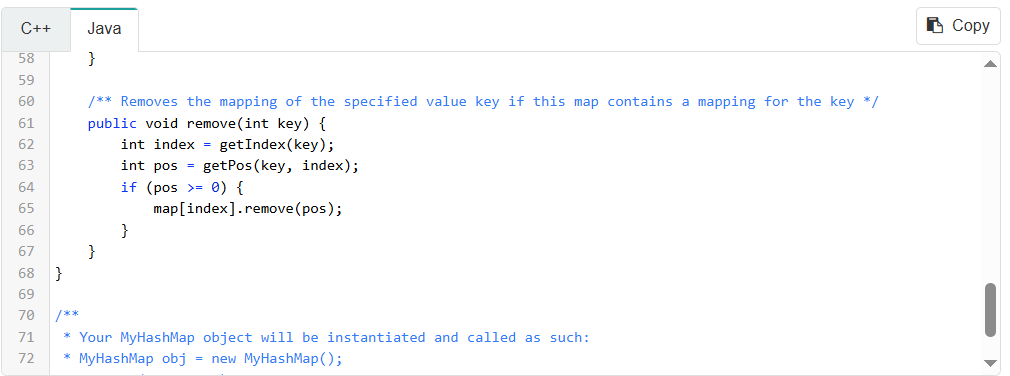
*Hash Map*











*More*

Let's take a look at the operation "remove". After we find out the position of the element, we need to remove the element from the array list.

Let's assume that we are going to remove the ith element and the size of the array list is n.

The strategy used in the built-in function is to move all the elements after ith element one position forward. That is to say, you have to move n - i times. So the time complexity to remove an element from an array list will be O(n).

Consider different value of i. In average, we will move ((n - 1) + (n - 2) + ... + 1 + 0) / n = (n - 1) / 2 times.

Hopefully, there are two solutions to reduce the time complexity from O(n) to O(1).

**1. Swap**

There is a tricky strategy we can use. First, swap the element which we want to remove with the last element in the bucket. Then remove the last element. By this way, we successfully remove the element in O(1) time complexity.

**2. Linked List**

Another way to achieve this goal is to use a linked list instead of an array list. By this way, we can remove the element in O(1) time complexity without modifying the order in the list.