Problem 705: Design HashSet

Problem Information

Difficulty: Easy
Acceptance Rate: 67.47%
Paid Only: No
Tags: Array, Hash Table, Linked List, Design, Hash Function

Problem Description

Design a HashSet without using any built-in hash table libraries.

Implement `MyHashSet` class:

* `void add(key)` Inserts the value `key` into the HashSet. * `bool contains(key)` Returns whether the value `key` exists in the HashSet or not. * `void remove(key)` Removes the value `key` in the HashSet. If `key` does not exist in the HashSet, do nothing.

Example 1:

Input ["MyHashSet", "add", "add", "contains", "contains", "add", "contains", "remove", "contains"] [[], [1], [2], [1], [3], [2], [2], [2], [2]] **Output** [null, null, null, true, false, null, true, null, false] **Explanation** MyHashSet myHashSet = new MyHashSet(); myHashSet.add(1); // set = [1] myHashSet.add(2); // set = [1, 2] myHashSet.contains(1); // return True myHashSet.contains(3); // return False, (not found) myHashSet.add(2); // set = [1, 2] myHashSet.contains(2); // return True myHashSet.remove(2); // set = [1] myHashSet.contains(2); // return False, (already removed)

Constraints:

* `0 <= key <= 106` * At most `104` calls will be made to `add`, `remove`, and `contains`.

Code Snippets

C++:

```
class MyHashSet {
public:
MyHashSet() {

}

void add(int key) {

}

void remove(int key) {

}

bool contains(int key) {

}
};

/**
* Your MyHashSet object will be instantiated and called as such:
* MyHashSet* obj = new MyHashSet();
* obj->add(key);
* obj->remove(key);
* bool param_3 = obj->contains(key);
*/
```

Java:

```
class MyHashSet {

public MyHashSet() {

}

public void add(int key) {

}

public void remove(int key) {

}
public boolean contains(int key) {

}
}

/**
* Your MyHashSet object will be instantiated and called as such:
* MyHashSet obj = new MyHashSet();
* obj.add(key);
* obj.remove(key);
* boolean param_3 = obj.contains(key);
*/
```

Python3:

```
class MyHashSet:

def __init__(self):


def add(self, key: int) -> None:


def remove(self, key: int) -> None:


def contains(self, key: int) -> bool:



# Your MyHashSet object will be instantiated and called as such:
# obj = MyHashSet()
# obj.add(key)
# obj.remove(key)
# param_3 = obj.contains(key)
```