

다음의 기능을 제공하는 해시맵을 디자인하라.

- `put(key, value)` : Insert a (key, value) pair into the HashMap. If the value already exists in the HashMap, update the value.
- `get(key)`: Returns the value to which the specified key is mapped, or -1 if this map contains no mapping for the key.
- `remove(key)` : Remove the mapping for the value key if this map contains the mapping for the key.

1.collections.defaultdict

class ListNode:

```
def __init__(self, key = None, value = None):
    self.key = key
    self.value = value
    self.next = None
```

class MyHashMap:

```
def __init__(self):
    """
    Initialize your data structure here.
    """
    self.size = 1000
    self.table = collections.defaultdict(ListNode)
```

```
def put(self, key: int, value: int) -> None:
    """
    value will always be non-negative.
    """
    index = key % self.size
    if self.table[index].key is not None:
        node = self.table[index]
        while node:
            if node.key == key:
                node.value = value
                return
            elif not node.next:
                node.next = ListNode(key, value)
                return
            node = node.next
    else:
        self.table[index] = ListNode(key, value)
    return
```

```
def get(self, key: int) -> int:
    """
    Returns the value to which the specified key is mapped, or -1 if this map contains no
    mapping for the key
    """
```

```

index = key % self.size
if self.table[index].key is not None:
    node = self.table[index]
    while node:
        if node.key == key:
            return node.value
        node = node.next
return -1

```

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

```

"""
Removes the mapping of the specified value key if this map contains a mapping for the key
"""
index = key % self.size
if self.table[index].key is not None:
    prev, node = None, self.table[index]
    while node:
        if node.key == key:
            if not prev:
                self.table[index] = ListNode() if not node.next else node.next
            return
            prev.next = node.next
            return
        prev, node = node, node.next
return

```

Your MyHashMap object will be instantiated and called as such:

```

# obj = MyHashMap()
# obj.put(key,value)
# param_2 = obj.get(key)
# obj.remove(key)

```

- defaultdict를 사용해서 오류를 줄이자
- None, 0 과같은 False같은값을 주의하자

2.기본 딕셔너리

class ListNode:

```

def __init__(self, key = None, value = None):
    self.key = key
    self.value = value
    self.next = None

```

class MyHashMap:

```

def __init__(self):
    """

```

Initialize your data structure here.

```
"""
```

```
self.size = 1000
```

```
self.table = {}
```

```
def _del_node(self, head: ListNode, key: int) -> ListNode:
```

```
    prev, node = None, head
```

```
    while node:
```

```
        if node.key == key:
```

```
            if not prev:
```

```
                head = node.next
```

```
            else:
```

```
                prev.next = node.next
```

```
            return head
```

```
        prev, node = node, node.next
```

```
    return head
```

```
def put(self, key: int, value: int) -> None:
```

```
    """
```

```
    value will always be non-negative.
```

```
    """
```

```
    index = key % self.size
```

```
    if index in self.table:
```

```
        node = self.table[index]
```

```
        while node:
```

```
            if node.key == key:
```

```
                node.value = value
```

```
            return
```

```
            elif not node.next:
```

```
                node.next = ListNode(key, value)
```

```
            return
```

```
            node = node.next
```

```
    else:
```

```
        self.table[index] = ListNode(key, value)
```

```
    return
```

```
def get(self, key: int) -> int:
```

```
    """
```

Returns the value to which the specified key is mapped, or -1 if this map contains no mapping for the key

```
    """
```

```
    index = key % self.size
```

```
    if index in self.table:
```

```
        node = self.table[index]
```

```
        while node:
```

```
            if node.key == key:
```

```
                return node.value
```

```
        node = node.next
    return -1
```

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

```
    """
    Removes the mapping of the specified value key if this map contains a mapping for the key
    """
    index = key % self.size
    if index in self.table:
        self.table[index] = self._del_node(self.table[index], key)
        if not self.table[index]:
            del self.table[index]
    return
```

Your MyHashMap object will be instantiated and called as such:

```
# obj = MyHashMap()
# obj.put(key,value)
# param_2 = obj.get(key)
# obj.remove(key)
```

3.딕셔너리

class MyHashMap:

```
    def __init__(self):
```

```
        """
        Initialize your data structure here.
        """
        self.data = {}
```

```
    def put(self, key: int, value: int) -> None:
```

```
        """
        value will always be non-negative.
        """
        self.data[key] = value
```

```
    def get(self, key: int) -> int:
```

```
        """
        Returns the value to which the specified key is mapped, or -1 if this map contains no
        mapping for the key
        """
        if key in self.data:
            return self.data[key]
        return -1
```

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

```
    """
```

```
    Removes the mapping of the specified value key if this map contains a mapping for the key
```

```
    """
```

```
    if key in self.data:
```

```
        del(self.data[key])
```

```
# Your MyHashMap object will be instantiated and called as such:
```

```
# obj = MyHashMap()
```

```
# obj.put(key,value)
```

```
# param_2 = obj.get(key)
```

```
# obj.remove(key)
```