Problem: Implement a hash table with set, get, and remove methods.

Constraints

- For simplicity, are the keys integers only?
    - Yes
- For collision resolution, can we use chaining?
    - Yes
- Do we have to worry about load factors?
    - No
- Do we have to validate inputs?
    - No
- Can we assume this fits memory?
    - Yes

Test Cases

- `get` no matching key -> KeyError exception
- `get` matching key -> value
- `set` no matching key -> new key, value
- `set` matching key -> update value
- `remove` no matching key -> KeyError exception
- `remove` matching key -> remove key, value

Algorithm

Hash Function

- Return key % table size

Complexity:

- Time: O(1)
- Space: O(1)

Set

- Get hash index for lookup
- If key exists, replace
- Else, add

Complexity:

- Time: O(1) average and best, O(n) worst

- **Space: O(1) space for newly added element**

Get

- **Get hash index for lookup**
- **If key exists, return value**
- **Else, raise KeyError**

Complexity:

- **Time: O(1) average and best, O(n) worst**
- **Space: O(1)**

Remove

- **Get hash index for lookup**
- **If key exists, delete the item**
- **Else, raise KeyError**

Complexity:

- **Time: O(1) average and best, O(n) worst**
- **Space: O(1)**