How would you implement a hash map from scratch?

A hash map is actually implemented using an **array**.
The key–value pair in a hash map gets converted into an integer using a **hashing function**.
We then use that integer as the **index** to determine where to place the key–value pair in the array.

When hashing, we typically assign a value to each character (determined by its **ASCII value**) and add the total.
We then take the total and compute the **modulus** with the length of the array.
This guarantees that the result is a valid index within the array.

What are some problems with hashing?

One common problem is **collisions**.
A collision occurs when two different key–value pairs produce the **same index** after hashing.
Collisions can be **minimized** but never completely eliminated.

How to minimize collisions:

1. **Monitor load factor** – Keep track of the size of the array and how many keys are inserted.
   When the hash map is **half full** (i.e., the number of keys is half the size of the array), resize the array.

2. **Resizing** – Similar to resizing a dynamic array:

   - Double the size of the hash map.

   - Copy all old values into the new array.

3. **Rehashing** –
   When resizing, you **cannot** simply copy key–value pairs to the same indices.
   Since the array size changes, the **hash results will change**.
   Therefore, you must **rehash** each key using the new array size to compute new indices.
   This helps reduce collisions, though it does not eliminate them entirely.

Strategies for handling collisions:

1. Separate Chaining (Linked List)

- Store a linked list of key–value pairs at each index.
- Downside: Retrieval may require searching through the entire linked list.

2. Open Addressing

- Loosely define where a key should be placed.
- If the index is occupied, check the **next index** (increment by 1) until you find an empty spot.
- Downside: This can lead to a naive approach of checking multiple indices in sequence.

Optimization Tip:

To reduce clustering in **open addressing**, make the array size a **prime number**.
When resizing, instead of exactly doubling the size, **increase to the closest prime number** greater than double the current size for better distribution.

If you want, I can also **turn this into a clean interview-ready cheat sheet** with diagrams showing collisions, open addressing, and separate chaining so it's more visual. That would make it easier to study and present.