# **Collisions**

**Collision Definition**

* When two or more element values in a hash table produce the same result from its hash function, indicating that they both prefer to be stored in the same index of the table.

**Why Do We Care?**

* Consider the problems caused by a collision.
* You can’t disallow insertions because that would make your hash table severely limited.
* You could even have a collision when inserting into a hash table that contains one item!
* Therefore, you need to find a new location for the data item that hasn’t already been taken.
* This is called **collision resolution**.

**Practical Example of Collision**

* We use the hash function

***h*(*x*) = *x* % *TableSize* where**

***x*** = hash function input

***TableSize*** = table size

* Suppose that you want to insert an item whose search key is 4567 into a hash table ***table*** of size 101.

***h*(*x*) =** 4567**%** 101

***h*(*x*) =** 22

* The hash function outputs 22
* This means that we place the new item at index 22 (***table***[22]).
* Suppose, however, that ***table***[22] already contains an item.
* Let’s say you had earlier placed 7597 into ***table***[22] because 7597 mod 101 equals 22.
* Where do you place this new item?
* You certainly do not want to disallow the insertion on the grounds that the dictionary is full.
* We need to find a new location for the data item that hasn’t already been taken.
* This is called **collision resolution**.

Diagram

Description automatically generated with medium confidence

# **Types of Collision Resolution**

* Two general approaches to collision resolution are common.

1. **Closed hashing (Open Addressing)** - new items are placed into another location.
   1. Linear probing
   2. Quadratic Probing
   3. Double Hashing
2. **Open hashing (Reconstructing the hash table) -** each location table[i] can accommodate more than one item.
   1. Separate chaining