**Aim**

To implement a hash table for storing integer keys, using linear probing to resolve collisions.

**Algorithm: Hash Table with Linear Probing**

1. **Initialize:**
   * Create an array hashTable of size TABLE\_SIZE.
   * Initialize all elements to a special value (e.g., -1) to indicate empty slots.
2. **Hash Function:**
   * Define a hash function: hash(key) = key % TABLE\_SIZE.
3. **Insertion:**
   * Compute index using hash function: index = hash(key).
   * If hashTable[index] is empty, insert the key there.
   * Else (collision occurs), perform **linear probing**:
     + Set i = 1.
     + While the slot at (index + i) % TABLE\_SIZE is occupied and i < TABLE\_SIZE, increment i.
     + Insert key at the first empty slot found.
   * If no empty slot is found after probing the entire table, report **hash table full**.
4. **Searching:**
   * Compute initial index: index = hash(key).
   * If hashTable[index] equals the key, return index.
   * Else, perform linear probing:
     + Set i = 1.
     + While hashTable[(index + i) % TABLE\_SIZE] is not empty and not equal to the key, increment i.
     + If key is found, return index; else, report **not found**.
5. **Display:**
   * Traverse the entire hashTable array.
   * Print each index and the stored key or indicate if the slot is empty.

