Assignment 3: Module 2 (1b) Report:

This program implements a hash table using quadratic probing for collision resolution.

TABLE_SIZE is defined as 11, indicating the size of the hash table. It's a prime number often chosen for hash tables to reduce clustering and improve distribution of keys.

- **Function Description**:
  - h1 computes the initial hash value (home_slot) for a given key.
  - It uses a series of arithmetic operations to ensure the hash value falls within the range [0, TABLE_SIZE - 1].

Function Description:

- quadratic_probe calculates the next slot to probe using quadratic probing.
- It uses the formula (home_slot + (k * k + k) / 2) % TABLE_SIZE to compute the next slot, where k is the probe sequence index.
- Quadratic probing helps to systematically search for an empty slot in case of collisions, reducing clustering compared to linear probing.

Function Description:

- initialize_hash_table initializes all slots of hash_table to -1, indicating that they are initially empty.
- **Function Description**:
  - print_hash_table prints the final contents of the hash table after all insertions.
  - It displays a formatted representation:
    - Headers for slots (Slot 0 to Slot 10).
    - Contents of each slot in the hash table. Empty slots are represented by spaces.

Function Description:

- insert_key inserts a key with corresponding value into the hash table using quadratic probing.
- **Steps**:
  1. Calculates the home_slot using the h1 hash function.
  2. Initializes k to 0 for quadratic probing.
  3. Calculates the initial slot using quadratic_probe(home_slot, k).
  4. Prints the key, home_slot, and sequence of probe positions until an empty slot (-1) is found.
  5. Inserts the key into the found slot.
- **Main Function Description**:
  - Initializes a hash_table of size TABLE_SIZE (11) using initialize_hash_table.

    - Calls insert_key multiple times to insert keys into the hash table.
    - Finally, calls print_hash_table to display the contents of the hash table after all insertions.

1. **Initialization**:
    - The hash_table is initialized with all slots set to -1 (empty).
2. **Insertions**:
    - Each insert_key call calculates the initial slot (home_slot) using h1.
    - It then uses quadratic probing (quadratic_probe) to find an empty slot and inserts the key.
3. **Output**:
    - During insertion, it prints the key being inserted, its home slot, and the sequence of probe positions.
    - After all insertions, it prints the final hash table, showing the contents of each slot.

This C program demonstrates the implementation of a hash table using quadratic probing for collision resolution. It utilizes a hash function (h1) to compute initial positions and a quadratic probing function to handle collisions. The program initializes, inserts keys, and prints the final state of the hash table, providing a clear example of how quadratic probing can be applied to resolve collisions efficiently in a hash table implementation.