Assignment 3: Module 2 (1a) Report:

This code implements a simple hash table using open addressing with linear probing for collision resolution.

Function Description:

- h1 is a hash function used to compute the index where an element (key) should be inserted into the hash table.
- It computes x using a series of arithmetic operations:
    1. (key + 7) * (key + 7): Squares key + 7.
    2. x = x / 16: Divides x by 16.
    3. x = x + key: Adds key to x.
    4. x = x % 11: Computes x modulo 11 to ensure the result fits within the range of the hash table (0 to 10).

- **Function Description**:
    - insert function inserts a key into the hash table using linear probing for collision handling.
    - **Steps**:
        i. home_slot = h1(key): Computes the initial slot (home_slot) using the hash function h1.
        ii. probe_sequence[11]: Array to store the sequence of probe positions.
        iii. while (hash_table[(home_slot + i) % 11] != 0): Loops until an empty slot (0 indicates empty in the hash table) is found.
            - Computes the actual slot using (home_slot + i) % 11.
            - Stores each position in probe_sequence.
        iv. Inserts key into the first empty slot found.
        v. Prints the key, home_slot, and probe_sequence for diagnostic purposes.

Function Description:

- print_hash_table function prints the final contents of the hash table after all insertions.
- Prints a formatted representation of the hash table:
    - Headers for slots (Slot 0 to Slot 10).
    - Contents of each slot in the hash table.
- **Main Function Description**:
    - Initializes a hash table hash_table of size 11 with all slots initially set to 0.
    - Calls insert function to insert several keys into the hash table.

- After all insertions, calls print_hash_table to display the final state of the hash table.

Explanation of the Example Execution

1. **Initial Setup**:
   - hash_table is initialized with size 11, all elements set to 0.
2. **Insertions**:
   - Each insert call calculates the initial slot (home_slot) using h1(key).
   - If the slot is occupied (hash_table[(home_slot + i) % 11] != 0), it probes sequentially ((home_slot + i) % 11) until an empty slot is found.
   - Prints the key, home_slot, and sequence of probe positions.
3. **Print Hash Table**:
   - Displays a formatted representation of the hash table with slots and their contents.

This C program demonstrates a basic implementation of a hash table using open addressing with linear probing for collision resolution. The hash function h1 is used to compute initial positions, and the insert function handles collisions by probing sequentially until an empty slot is found. The print_hash_table function is used to visualize the final state of the hash table after all insertions. This example provides insight into how hash tables work and how linear probing can be implemented to handle collisions efficiently.