Assignment 3: Module 2 (1c) Report:

This program implements a hash table using double hashing for collision resolution. Here's a detailed explanation of each part:

TABLE_SIZE is defined as 11, which determines the size of the hash table. It's a prime number commonly used to reduce clustering and improve the distribution of keys.

Function Description:

- h1 computes the initial hash value (home_slot) for a given key using modulo operation.
- This function calculates the initial slot based on the remainder of key divided by TABLE_SIZE.

Function Description:

- reverse function reverses the digits of the key.
- It repeatedly extracts the last digit of key, appends it to reversed, and then removes the last digit from key by integer division.
- For example, reverse(123) would return 321.

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 double hashing.
- **Steps**:
  1. Calculates the home_slot using the h1 hash function.
  2. Calculates the initial probe using the reverse function on the key.
  3. Initializes slot to home_slot.

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.

Initialization:

- The hash_table is initialized with all slots set to -1 (empty).

Insertions:

- Each insert_key call calculates the initial slot (home_slot) using h1.
- It calculates the initial probe (reverse(key)) and then uses double hashing to find an empty slot.
- During insertion, it prints the key, its home slot, and the sequence of probe positions until it finds an empty slot.

Output:

- 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 double hashing for collision resolution. It utilizes a simple hash function (h1) for initial slot calculation and a reverse function (reverse) for probing sequence calculation. The program initializes, inserts keys, and prints the final state of the hash table, providing a clear example of how double hashing can be applied to resolve collisions efficiently in a hash table implementation.