The provided C code demonstrates how to implement a hash table with linear probing to resolve collisions. Here's a brief explanation of each part:

1. **Hash Function (h1)**:
   - The function `h1` computes the hash value for a given key. It first adds 7 to the key, squares the result, divides by 16, adds the original key, and then takes the result modulo the table size (11) to get the final hash value, which determines the home slot.

2. **Insertion with Linear Probing**:
   - The function insert Linear Probing takes the hash table and a key as inputs. It calculates the home slot using the hash function. If the home slot is occupied, it searches for the next available slot by checking subsequent slots in a linear

manner until it finds an empty one or comes full circle to the home slot, indicating the table is full.

3. **Initialization and Table Printing**:
   - The `main` function initializes the hash table with -1, indicating empty slots. It then inserts a predefined list of keys into the table using the `  Probing` function. After inserting all keys, it calls print Hash Table to display the contents of the table.
   - The `print Hash Table` function iterates through the table and prints each slot number along with its contents. If a slot is empty, it prints a space.

This implementation effectively demonstrates the use of linear probing to handle collisions in a hash table.