

## Hash Function h1

The h1 function takes an integer key as input and returns a hash value between 0 and M-1 (where M is defined as 11). The hash function is a simple quadratic function that combines the key with a constant offset and scaling factor.

Here's what the hash function does:

1.  $x = (\text{key} + 7) * (\text{key} + 7)$ : This line calculates a quadratic value based on the key. The + 7 is an arbitrary offset to avoid collisions.
2.  $x = x / 16$ : This line scales down the quadratic value to reduce the range of the hash values.
3.  $x = x + \text{key}$ : This line adds the original key to the scaled quadratic value to introduce more variability in the hash values.
4.  $x = x \% M$ : This line takes the result modulo M to ensure the hash value falls within the range of 0 to M-1.

The resulting hash value is used to determine the initial slot in the hash table where the key should be stored.

## Insert Function :

The insert function takes a key and a hash\_table as input. It calculates the home slot for the key using the h1 hash function and inserts the key into the hash table using linear probing.

Here's what the insert function does:

1. `int home_slot = h1(key)`: This line calculates the home slot for the key using the h1 hash function.
2. `int probe_slot = home_slot`: This line initializes the probe slot to the home slot.
3. `while (hash_table[probe_slot] != -1)`: This loop continues until an empty slot is found.
4. `probe_slot = (probe_slot + 1) % M`: This line increments the probe slot by 1 and wraps around to 0 if it reaches the end of the hash table.
5. `if (probe_slot == home_slot)`: This line checks if the probe slot has wrapped around to the home slot, indicating that the hash table is full.
6. `printf("Hash table is full\n")`: If the hash table is full, this line prints an error message.
7. `return`: If the hash table is full, this line returns from the function.
8. `hash_table[probe_slot] = key`: If an empty slot is found, this line inserts the key into the hash table.

## Print Function print\_hash\_table

The print\_hash\_table function takes a hash\_table as input and prints its contents in a formatted table.

Print function:

1. `printf("Slot ")`: This line prints the header for the slot column.
2. `for (i = 0; i < M; i++)`: This loop iterates over the hash table slots.

3. `printf("%2d ", i)`: This line prints the slot number with a width of 2 characters.
4. `printf("\n")`: This line prints a newline character.
5. `printf("Contents ")`: This line prints the header for the contents column.
6. `for (i = 0; i < M; i++)`: This loop iterates over the hash table slots again.
7. `printf("%2d ", hash_table[i])`: This line prints the contents of each slot with a width of 2 characters.
8. `printf("\n")`: This line prints a newline character.

#### Main Function :

The main function initializes an empty hash table with M slots, each set to -1. It then inserts 9 keys into the hash table using the insert function and prints the resulting hash table using the `print_hash_table` function.

Here's what the main function does:

1. `int hash_table[M]`: This line declares an array of M integers to represent the hash table.
2. `for (i = 0; i < M; i++)`: This loop initializes each slot in the hash table to -1.
3. `insert(43, hash_table)`: This line inserts the key 43 into the hash table.
4. ... (similar insert statements for other keys)
5. `print_hash_table(hash_table)`: This line prints the resulting hash table.