This code demonstrates the implementation of hash tables using two methods: chaining and quadratic probing.

**Chaining:** In this approach, each position in the hash table is a linked list. When a collision occurs (multiple keys hash to the same index), the new key-value pair is added to the linked list at that index. The `insertChaining` function inserts key-value pairs, and the `searchChaining` function retrieves values by traversing the linked list. The `printChainingHashTable` function prints the hash table.

**Quadratic Probing:** This method resolves collisions by probing for the next available slot using a quadratic function (i.e., checking positions hashIndex + i^2). If a collision occurs, the algorithm checks the next slot in a quadratic sequence until

an empty slot is found. The `insertProbing` function inserts key-value pairs using this probing method, and the `searchProbing` function searches for values in a similar manner. The `printProbingHashTable` function prints the hash table.

In the `main` function, examples of using both chaining and quadratic probing are demonstrated by inserting and searching for keys, as well as printing the hash tables.