**Basic Image Detection with HashTable**

**Objective:**

This homework introduces fundamental concepts in image detection and hashing through the implementation of a simplified image detection system. You will handle binary images (28x28 grids) stored in text files, convert them into Run Length Encoded (RLE) strings, and manage them in a hash table for efficient storage and retrieval.

**Key Tasks:**

1. **Image Representation**:
   * Images are provided as binary text files (imageX.txt) containing 28x28 grids of 0s (black) and 1s (white).
   * Convert these binary images into a single flattened string, then compress it using **Run Length Encoding (RLE)**.
2. **Run Length Encoding (RLE)**:
   * Compresses the binary string by replacing consecutive 0s or 1s with a count and a label (B for black, W for white).
   * Example: 0000001111 becomes 6B4W.
3. **Hash Table Implementation**:
   * Store RLE strings in a hash table using a custom hash function.
   * Handle hash collisions via **linear probing**.
   * Ensure each RLE string is unique and can be retrieved accurately, even if collisions occur.
4. **Query Processing**:
   * Query images (queryX.txt) are also encoded using RLE and compared against the hash table.
   * If a match is found, display the decoded image.
   * If no match is found, print the RLE of the query and notify the user.

**Program Flow:**

1. **Image Insertion**:
   * The program asks for image file numbers to insert into the hash table.
   * Input "query" to switch to the querying stage.
2. **Query Processing**:
   * Query images to check if they exist in the hash table.
   * Input "exit" to terminate the program.

**Output:**

* If a match is found: Display the stored RLE and its decoded 28x28 binary image.
* If no match: Display the RLE of the query and indicate no match.

**Submission Guidelines:**

* Submit HashTable.h, HashTable.cpp, and a test file.
* Ensure compatibility with **CodeRunner** on SUCourse.
* Late or non-SUCourse submissions will not be accepted.