BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSE208 - Data Structures and Algorithm Sessional

Implementation of Hash Table Data Structure

1 Problem Specification

In this assignment, you are required to implement a hash table that satisfies the following specifications:

1. The hash table must support insertion, search, and deletion operations. The table size shall be dynamically adjusted based on the load factor and must always remain a prime number. The initial table size may be set to 13, and the cut-off load factor to 0.5. These parameters must be defined using single-source variables to facilitate modification during evaluation.

   If the load factor exceeds 0.5, the table size should be expanded to the smallest prime number greater than twice the current size. Conversely, if the load factor falls below 0.25, the table size should be reduced to the largest prime number smaller than half of the current size. Table compaction should not be performed when the table size is equal to the initial size. Furthermore, an expansion operation must occur only after at least $n/2$ insertions since the previous expansion, and a compaction operation must occur only after at least $n/2$ deletions since the previous compaction, where n denotes the number of elements present at the time of the corresponding resizing operation.

2. The hash table shall store data in the form of key–value pairs. Both keys and values may be of arbitrary data types.

3. For demonstration purposes, randomly generated unique words of length n (where n is provided by the user) must be inserted into the hash table. These words may be meaningful or meaningless. To support this requirement, a random word generator must be implemented. The generated words will serve as the keys, and insertion must be performed sequentially. Any duplicate words generated must be discarded. The sequence number corresponding to the generation order of each unique word will be used as the associated value.

2 Example

Assume that the word generator produces the following five words:

ancient
puzzled
benefit
ancient
zigzags

The resulting key–value pairs will be:

(ancient, 1)
(puzzled, 2)
(benefit, 3)
(zigzags, 4)

The second occurrence of *ancient* is discarded because the word already exists in the hash table.

3 Hash Function

You are required to use two standard hash functions, denoted as Hash1(k) and Hash2(k), selected either by design or from well-established literature. The hash functions should be chosen with the objective of minimizing collisions. It is expected that at least 60% of the keys map to unique hash values (i.e., at least 6000 unique hash values for 10000 entries).

4 Collision Resolution

To resolve collisions, the following three techniques must be implemented:

4.1 Chaining Method

All elements that hash to the same index are stored in a linked list. Each slot j in the hash table contains a pointer to the head of the list comprising all elements whose keys hash to j. If no such elements exist, slot j contains `NULL`.

4.2 Double Hashing

For double hashing, the probe sequence is defined as:

$$\text{doubleHash}(k, i) = (\text{Hash}(k) + i \times \text{auxHash}(k)) \bmod N$$

where Hash(k) denotes one of the primary hash functions described in Section 3 (both Hash1 and Hash2 must be evaluated as described in Section 5), and auxHash(k)

represents an auxiliary hash function. For simplicity, the auxiliary hash function may be kept straightforward. The initial probe targets Table[Hash(k)], and each subsequent probe advances by auxHash(k) positions modulo N.

4.3 Custom Probing

In the case of custom probing, the probe sequence is defined as:

$$\text{customHash}(k, i) = (\text{Hash}(k) + C_1 \times i \times \text{auxHash}(k) + C_2 \times i^2) \bmod N$$

where C_1 and C_2 are user-defined constants. All other probing mechanisms should follow the same principles as those used in double hashing.

5 Report Generation

For performance evaluation, generate 10,000 unique words of length $n = 10$ using the previously described method and insert them into the hash table. Record the total number of collisions for each collision resolution technique using both Hash1 and Hash2, and present the results in a tabular format. From the same dataset, randomly select 1,000 words and search for each word in the hash table. Compute and report the average number of hits required per search operation. This evaluation must also be conducted separately for both hash functions. Here, the number of collisions refers to the number of instances in which a hash function maps distinct keys to the same hash value. The number of hits corresponds to the total number of table accesses required to successfully locate a key during a search operation.

The report should be presented in the following tabular format:

	Hash1		Hash2	
	Number of Collisions	Average Hits	Number of Collisions	Average Hits
Chaining Method				
Double Hashing				
Custom Probing				

6 Submission Deadline

The submission deadline is **11:59 PM on 1 February 2025 (Sunday)**.