Project Description

Title: Implementation of a Separate Chaining Hash Symbol Table

Overview:
In this project, students will implement the Separate Chaining Hash API using separate chaining for collision resolution. The hash table is backed by an array of sequential search symbol tables (linked lists) where each chain stores key-value pairs. The goal is to understand how hash tables handle collisions, perform dynamic resizing, and provide basic symbol table operations.

Objectives:

- **Implement Core API Methods:**
  - `put(key, value)` : Insert a new key-value pair or update an existing key's value.
  - `get(key)` : Retrieve the value associated with a given key.
  - `delete(key)` : Remove a key (and its value) from the table.
  - `contains(key)` : Check if the table contains a given key.
  - `keys()` : Return an iterable collection of all keys.
  - `size()` and `isEmpty()` : Report the number of key-value pairs and whether the table is empty.
- **Dynamic Resizing:**
  - Expand the table (e.g., double the chains) when the average chain length is too high (10 or more).
  - Shrink the table (e.g., half the chains) when the average chain length becomes too low (2 or fewer) while keeping a minimum capacity.