Lab 3

Github Link

https://github.com/rusuraluca/lftc/tree/main/lftc_lab2

Docs

`class Node` for storing key-value pairs in the hash table

- `key`: number
  - key associated with the value
- `value`
  - value associated with the key
- `next`
  - reference to the next `Node` object in the linked list
  - used for handling collisions in the hash table

`class HashTable` for representing the hash table data structure

- `capacity`: number
  - current capacity of the hash table, which starts with an initial capacity of 2
- `elmnt_no`: number
  - number of elements (key-value pairs) stored in the hash table
- `elmnt_list` = array<Node>
  - list used to store elements, with each index potentially holding a linked list of Node objects

`init`

- we initialize a hash table and set the `capacity` initially to 2, `elmnt_no` is 0 and `elmnt_list` is formed of 2 positions of None

`hash(value)`

- calculates the hash code for a given value (key)
  - if the value is an integer, it calculates the hash code as the remainder of the integer when divided by the current capacity
  - if the value is a string, it calculates the hash code as the sum of the ASCII values of its characters, divided by the current capacity

`insert(key, value)`

- to insert a key-value pair into the hash table
  - if the load factor (ratio of elements to capacity) is greater than or equal to 2, it triggers a resize and rehash operation
  - it calculates the hash for the key and inserts the new `Node` into the appropriate position in the linked list at that index

`get(key)`

- retrieves the value associated with a given key
- it calculates the hash for the key, searches the linked list at that index, and returns the value if the key is found, if not found, it returns `None`

`resize_and_rehash()`

- called when the load factor exceeds 2
  - doubles the hash table capacity
  - it creates a deep copy of the existing element list, resets the element list with the new capacity, and reinserts the elements using the `insert` method