Github repository: <https://github.com/Dospinescu-Rares/FLCD-Project>

FLCD week 3 assignment

Dospinescu Rares

Group 932/1

Informatics in English Specialization

***class SymbolTable:***

The SymbolTable class provides a simple interface to manage symbols (identifiers and constants) using a custom hash table implementation. It allows adding, removing, checking existence, and finding the position of symbols based on their keys.

**def add(self, key):**

Adds a key to the SymbolTable

* **key**: The key being inserted
* **return**: The position in which the key was inserted

**def remove(self, key):**

Removes a key from the SymbolTable

* **key**: The key being removed
* **return:** None

**def contains(self, key):**

Checks for the presence of a key in the SymbolTable

* **key**: The key we are searching for
* **return:** The location of the key, -1 otherwise

**def get\_position(self, key):**

Searches for the key’s position in the SymbolTable

* **key**: The key whose position we are searching for
* **return:** The position of the key

**def \_\_str\_\_(self) -> str:**

* **return:** The SymbolTable as a string

**class HashTable:**

Custom implementation of a hash table data structure. This hash table allows storing keys and provides basic operations like hash, add, remove, contains and get\_position.

* **size:** The size of the HashTable

**def hash(self, key):**

Returns the hash of a given key

* **key:** The key whose hash we are looking for
* **return:** The hash of a key

**def add(self, key):**

Adds a key to the HashTable

* **key:** The key being added to the HashTable
* **return:** The position in which the key was inserted

**def remove(self, key):**

Removes a key from the HashTable

* **key:** The key being removed from the HashTable
* **return:** None

**def contains(self, key):**

Searches for the key in the HashTable

* **key:** The key we are searching for
* **return: True if the key is part of the HashTable, False otherwise**

**def get\_position(self, key):**

Finds the position of a key in the HashTable

* **key:** The key we are searching for
* **returnn:** The position and the index of the deque where the key is located

**def \_\_str\_\_(self):**

* **return:** Returns the HashTable as a string