https://github.com/921-Beltechi-Lois/Formal-Languges-and-Compiler-Design

Documentation
Symbol Table will be composed of 3 separate hash tables:
-one for identifiers
-one for string constants
-one for integer constants

Each Hashtable is represented by an array in which every position is another array because we want to store values that hash to the same position.
Hashtables have a size.

An element from Symbol Table has as position a pair of indices:
-first one being the index of the list in which the element is stored;
-the second one being its actual position in that list

Hash function will be:
-value % size of the array/list, for integer values
- (characters ?) % (size of the array/list), for string constants / identifiers
-sum of the ASCII codes of the characters modulo the size of the list, for string constants/identifiers

Symbol table
- addIdentifier(name: string): (int, int) – add an identifier and return its position in the ST
- addIntConstant(constant: int): (int, int) – add an integer constant and return its position in the ST
- addStringConstant(constant: string): (int, int) – add an string constant and return its position in the ST
- getPositionIdentifier(name: string): (int, int) – get the position of the identifier in the ST
- getPositionIntConstant(constant: int): (int, int) – get the position of the integer constant in the ST
- getPositionStringConstant(constant: string): (int, int) – get the position of the string constant in the ST
- toString() (overridden method) – get the string content of the whole symbol table

Hash table
- hash(key: int): int – gets the position in the ST of the list in which the integer constant will be added
- hash(key: string): int – gets the position in the ST of the list in which the string constant/identifier will be added, based on the sum of the ASCII codes of their characters
- getSize(): int – return the size of the hash table
- getHashValue(key: T): int – return the position in the ST according to the type of the parameter ‘key’

- add(key: T): (int, int) – add the key to the hash table and return its position if the operation is successful; if not, throw an exception
- contains(key: T): boolean – return if the given key is in the hash table or not
- **getPosition**(key: T): (int, int) – return the position in the ST of the given key, if it exists, if not, return (-1, -1)
- toString() (overridden method) – return the string content of the hash table