<https://github.com/BodaViktoria/5_FLCD/tree/master/Lab_3_final>

SymbolTable  
  
Representation:  
The symbol table contains, in its representation, a list and the size of the list, in which the elements will be place. At each position of the table there is a list of elements, since if two different constants/variables are hashed to the same symbol table index, they all can be stored. Thus, an element which is included in the symbol table, is represented by two indices, one showing the index of the list in which the element is stored, and the second one shows the position of the element in the specific list. I chose to implement the symbol table with the data structure hash table. The hash function consists of the sum of the ASCII codes of the characters of the constant/identifier, modulo the size of the list.  
  
Operations:  
- insert(key:string): -

Checks if the element was already stored in the ST or not. If it was, then the element won't be stored again and it raises an Exception saying that the element already exists in the ST. Otherwise, the element is stored.  
- remove(key): None

Checks if the key is present in the ST. If yes, then it removes the key from the data structure. Otherwise, it raises an error.  
- key\_exists(key:string): boolean

Checks if the given key exists in the ST. If it exists, then it returns True, otherwise it returns False.  
- hash(key): int

Computes the sum of the ASCII codes of the characters and divides them by the hash table size, then it returns it. The returned number is the index of the ST list in which the element will be placed.  
- \_\_str\_\_ (overridden operation): string

Returns the string representation of the ST.