ADTs Hash Table.

HashTable = {size=<size>, table=<table>}
Table =<Node1, Node2, Node3, Node_n>
Node = <K, V, Previous, Next>

$inv: HT.table.length = HT.size \land \forall x, y \in HT.table, x \neq y \Rightarrow hashFunction(x) \neq hashFunction(y) \land k \in (String \lor R)$

Primitive Operations:

● HashTable: <size>		$\longrightarrow$ HashTable
● hashFunction	Key	$\longrightarrow$ Integer
● getValue	HashTable x Key	$\longrightarrow$ Value
● FindValue	HashTable x Key	$\longrightarrow$ Value
● add	HashTable x Key x Value	$\longrightarrow$ HashTable
● delete	HashTable x Key x Value	$\longrightarrow$ HashTable

HashTable()
"Creates a new HashTable"
$\{pre: True \land k \in (String \lor R)$

{ post: HashTable = {table = <table>} } A new hash table is instantiated

hashFunction(K)
"Calculates the index of a given key in the hash table"

{ $k \in (String \lor R)$ }

{ post: non-negative integer less than the size of the hash table is given }

add(HashTable, k, v)
"Adds a new node to the hash table, in a specific position given by the hash function"

{ pre: TRUE $k \in (String \lor R)$}

{post: If there is no collision, Table<newNode, …, …>a new node is added to the table, in the index given by the hashFunction. If there is a collision, the new node is added at the end of the double linked list of index: Table<Node->newNode, …, …> . }

get(HashTable, K)
"Returns the first value associated with the given key in its corresponding index"

$\{pre\colon\ True,\ k \in (String\ \vee\ R)\}$

{ post: $< Value >$ Returns the value associated with the given key or null if it is not found }

find(HashTable, K)
"Returns the first value associated with the given key in the index, in case that there are collisions: more than one node stored in an index"

$\{pre\colon\ True,\ k \in (String\ \vee\ R)\}$

{ post: $< Value >$ Returns the value associated with the given key or null if it is not found }

delete(HashTable, K)
"Removes the node associated with the given key from the hash table"

$\{pre\colon\ True,\ k \in (String\ \vee\ R) \wedge node\ searched\ to\ remove\ exists\ in\ the\ hash\ table\}$

{ post: The node is correctly deleted from the hash Table }