TAD DoublyLinkedList

DoublyLinkedList= {Head = <head>, Head.Next=<next>, Head.Prev=<prev>, Value = <value>}

Invariant: Head != empty, Next != empty, Prev != ≡ ∧ (Next.value ≤ Head.value || Next.value ≥ Head.value) ∧ (Prev.value ≤ Next.value || Prev.value ≥ Next.value)

Construction operations:
*Create : → DoublyLinkedList
Modifier operations:
*addElement: DoublyLinkedListxValue → DoublyLinkedList
*remove: DoublyLinkedListxValue → DoublyLinkedList
Operaciones analizadoras:
*isEmpty: DoublyLinkedList → booleano
*size: DoublyLinkedList → Integer
*search: DoublyLinkedList → DoublyLinkedList

Create (value)
"Creates an element of the Doubly linked list with the prev and nect elements empty, but with a defined value"

{pre: TRUE }

{post: elementDoublyLinkedList = {Next = <nill>, Prev = <nill>, Value=<value>}

addElement (value)
"Inserts an element on the Doubly linked list"

{pre: TRUE }

{post: head = {Next=<nill>, Prev=<nill>, Value=<value>}}

remove(value)
"Removes an element passed by parameter from the linked list"

{pre: element to be deleted is in the doubly linked list }

{post: False if the element wasn't found, True otherwise}

isEmpty(DoublyLinkedList):

"Informs if the doubly linked list is empty."

{pre: TRUE}
{pre: DoublyLinkedList={Head:<head>,...}

{post: False if the DoublyLinkedList.head!= nil, True otherwise}

size(DoublyLinkedList):

"Returns an Integer that represents the number of elements currently inserted in the doubly linked list."

{pre: TRUE}
{pre: DoublyLinkedList={Head:<head>,...}

{post: n | n ∈ Z+}

search(DoublyLinkedListxelement):

"Informs if there's a element in the doubly linked list based on an index passed by parameter."

{pre: DoublyLinkedListxelement }

{post: FALSE if the element is not in the doubly linked list, TRUE otherwise.}