TAD FOR QUEUE

TAD QUEUE

An one-tuple <E>, which is going to be saved on the queue

The first one-tuple which entered to the queue is the first in being removed.

Primitive operations:

CreateQueue:	element	Queue
EnQueue:	element	true
Front:	true	element
DeQueue:	true	element
isEmpty:	true	Boolean
size:	true	integer

CreateQueue(E)

Constructor operation

“creates a queue”

Pre: True

Pos: a new queue

Enqueue(E)

Modifier operation

“adds a new element to the queue”

Pre: the queue has been created

Pos: a new element added to the queue

Front()

Analyzer operation

“returns the first element of the queue”

Pre: the queue has been created

Pos: returns the first element in the queue or NIL

DeQueue()

modifier operation

“removes the first element from the queue and returns it”

Pre: the queue has been created

Pos: a element has been removed from the queue

IsEmpty ()

analyzer operation

“evaluates if the queue is either empty or not”

Pre: the queue has been created

Pos: True if the queue is empty, false if the queue is not empty

size ()

Analyzer operation

“returns the size of the queue”

Pre: the queue has been created

Pos: returns the size of the queue

TAD FOR STACK

TAD STACK

An one-tuple <E>, which is going to be saved on the stack

The last one-tuple which entered to the stack is the first in being removed.

Primitive operations:

CreateStack: Stack	element
Push: true	element
Top: element	true
Pop: element	true
isEmpty: Boolean	true
size: integer	true

CreateStack(E)

Constructor operation

“creates a stack”

Pre: True

Pos: a new stack

Push(v)

Modifier operation

“adds a new element to the stack”

Pre: the stack has been created

Pos: a new element added to the stack

Top()

Analyzer operation

“returns the first element of the stack”

Pre: the queue has been created

Pos: returns the last element in the stack or NIL

pop()

modifier operation

“removes the last element from the stack and returns it”

Pre: the queue has been created

Pos: a element has been removed from de stack

IsEmpty () analyzer operation “evaluates if the stack is either empty or not” Pre: the stack has been created Pos: True if the stack is empty, false if the stack is not empty

size () Analyzer operation “returns the size of the stack” Pre: the stack has been created Pos: returns the size of the stack

----

TAD FOR HASHTABLE

TAD HASHTABLE

An two-tuple <K,E>, where E is an element which is going to be saved on the hash table and K is its key, used to select the slot where E is going to be saved. The key is saved along its Element.

hashtable uses the key to assign and found the slot where the element is going to be saved, remover or just searched

Primitive operations:

CreateHashTable: element
hashtable

TableInsert: element
true

TableRetrieve: key
element

TableDelete: key
element

isEmpty: true
Boolean

TableLength: true
integer

HashFunction   key
integer

CreateHashTable(E)

construct operation

“creates a new hashtable”

Pre: true

Pos: a new hash table

HashInsert(E)

modifier operation

“inserts a new element to the hashtable, using its key to found the slot where the element is going to be saved”

Pre: the hash table has been created

Pos:a new element has been inserted to the hashtable

HashRetrieve(K)

analyzer operation

“searches an element with a given search key in the hash table”

Pre: the hash table has been created

Pos: returns an Element or NIL

HashDelete(K)

modifier operation

“deletes and element from the hash table using its key, and return the element, and left an element with key -1 and E NIl”

Pre: the hash table has been created

Pos:an element has been deleted from the hashtable and in its place is an element with key -1 and E NIL isEmpty()

analyzer operation

“evaluates if the hashtable is either empty or not”

Pre: the hashtable has been created

Pos: True if the hashtable is empty, false if the hashtable is not empty

TableLength()

Analyzer operation

“returns the length of the hashtable”

Pre: the hashtable has been created

Pos: returns the length of the hashtable

TAD FOR PRIORITYQUEUE

TAD PRIORITYQUEUE

An one-tuple <E>, which is going to be saved on the queue and has a priority.
the element with the most priority is the first in being removed
Primitive operations: CreateQueue: element Queue EnQueue: element true Front: true element DeQueue: true element isEmpty: true Boolean size: true integer

CreateQueue(E)

Constructor operation

“creates a queue”

Pre: True

Pos: a new queue

Enqueue(E)

Modifier operation

“adds a new element to the queue”

Pre: the queue has been created

Pos: a new element added to the queue

Front()

Analyzer operation

“returns the element the most priority of the queue”

Pre: the queue has been created

Pos: returns the first element in the queue or NIL

DeQueue()

modifier operation

“removes the element with the most priority from the queue and returns it”

Pre: the queue has been created

Pos: a element has been removed from the queue

IsEmpty ()

analyzer operation

“evaluates if the queue is either empty or not”

Pre: the queue has been created

Pos: True if the queue is empty, false if the queue is not empty

size ()

Analyzer operation

“returns the size of the queue”

Pre: the queue has been created

Pos: returns the size of the queue