Seguimiento 4

Realizado por: Tomás Ossa (A00372231) y Daniela Bonilla (A00372534)

TAD Lista Enlazada Ordenada

Lista Enlazada Ordenada={Nodo=<Nodo>}

{inv: ListaEnlazada.Nodo.next>=ListaEnlazada.Nodo.prev ⇔ (si y solo si) ListaEnlazada.Nodo.prev != null }

Operaciones primitivas:

Nodo= Elemento perteneciente a la Lista Enlazada ordenada
Lista Enlazada Ordenada= L.E.O

Métodos	Tipo Operación	Entradas	Salida
CrearNodo:	(Constructora)		→Nodo
InsertarNodo Ordenadamente:	(Modificadora)	Nodo x Nodo x Nodo	
EstaVacía:	(Analizadora)		→ Booleana
ExisteNodo:	(Analizadora)	Nodo x Nodo	→ Booleana
RemoverNodo:	(Modificadora)	Nodo	
ObtenerNodo:	(Analizadora)	Nodo	→ Nodo

createNode - CrearNodo() :

Crear un nuevo nodo

{pre:TRUE}

{post: nodo creado. Si la lista leo está vacía entonces el primer nodo de la lista es n}

insertNode - InsertarNodoOrdenadamente(n, nPrev, nNext)

*Se inserta el Nodo n dentro de la lista enlazada ordenada de forma que su nodo siguiente(nNext) (si existe) sea estrictamente mayor y su elemento anterior(nPrev) (si existe) sea menor o igual. El Nodo actual pasa a ser el Nodo insertado. *

{pre: leo está inicializada}

{pre: n existe (es diferente de null, ya que debe ser comparada de alguna manera) }
{post: El tamaño de la lista leo aumenta en uno}

isEmpty - EstaVacia ()
Verificar si la lista enlazada ordenada contiene elementos (nodos) o no
{pre:Lista enlazada ordenada exista}
{post: `True` si la L.E.O == null `False` de lo contrario }

existingNode - ExisteNodo (n, actualNode)
Verifica la existencia de un Nodo n en la lista leo, la variable actualNode sirve para navegar entre los nodos de la lista y compararla con n hasta que sea igual
{pre:Lista enlazada ordenada exista} {pre: n existe (es diferente de null, ya que no se puede buscar algo nulo) }
{post: `False` si n no existe en leo `True` si n existe en leo}

deleteNode - RemoverNodo (n)
Se elimina el elemento actual de la lista siempre y cuando exista, esta verificación se hace con el método ExisteNodo(). El nodo anterior al eliminado pasa a tener las relaciones de siguiente y anterior que tenía el nodo eliminado.
{pre: lista enlazada ordenada está inicializada} {pre: n existe (es diferente de null, ya que debe ser comparada de alguna manera) } {pre: n existe en la lista leo }
{post: Nodo eliminado}

getNode - ObtenerNodo (n)
Se obtiene un Nodo n de la lista siempre y cuando exista, esta verificación se hace con el método ExisteNodo().
{pre:lista ordenada está inicializada} {pre: n existe (es diferente de null, ya que debe ser comparada de alguna manera) } {pre: n existe en la lista leo }
{post:Retorna el nodo a obtener}