TAD Vector<a>

**Igualdad Observacional**

Si a y b son dos vectores, a es igual a b si se cumple una de las siguentes condiciones:

-a es vacia y b es vacia

-tienen los mismos elementos ordenados en las mismas posiciones

Usa

Natural,Bool,Secunecia<a>

**Parametro formal**

a

**Observadores básicos**

tamaño(Vector<a>)

ver(Vector<a>)

obtener(Vector<a>,posición)

esta(Vector<a>,elemento)

**Generadores**

vacio → Vector<a> {Post; Esta vacio}

a\_partir\_de(Secuencia<a>) → Vector<a>

{Post:El vector contiene los elementos de la secuncia ingresada}

**Otras operaciones**

agregar(vector<a>,elemento)

eliminar (Vector<a>,elemento)

**Axiomas**

agregar(vector<a>,elemento) :se agregar el elemento que queramos a ingresar en el vector

eliminar (Vector<a>,elemento) :se eliminará el elemento del vector que indiquemos,(pos:en el caso de que este vacio, nos devolvera que esta vacio )

vacio → Vector<a> {Post; Esta vacio}:se creera el vector vacío, sin ningún elemento

a\_partir\_de(Secuencia<a>) → Vector<a>:se creerá un vector a partir de la secuencia que pasemos como parámetro

{Post:El vector contiene los elementos de la secuencia ingresada}

tamaño(Vector<a>):devolverá la cantidad de elementos que tiene el vector

ver(Vector<a>) :nos mostrará los elementos que tiene el vector

obtener(Vector<a>,posicion):nos devolverá el elemento que está en la posición que le indicamos

**Exporta**

Vector<a>,agregar,eliminar,tamaño,ver,obtener,bool,

TAD Conjunto<a>

**Igualdad Observacional**

Si a y b son dos conjuntos, a es igual a b si se cumple una de las siguentes condiciones:

-a es vacia y b es vacia

-tienen los mismos elementos

Usa

Natural,Bool,Secunecia<a>

**Parametro formal**

a

**Observadores básicos**

tamaño(Conjunto<a>)

ver(Conjunto<a>)

esta(Conjunto<a>,elemento) Pos{devuelve un booleano}

**Generadores**

vacio → Conjuntor<a> {Post; Esta vacio}

a\_partir\_de(Secuencia<a>) → Vector<a>

{Post:El vector contiene los elementos de la secuencia ingresada,si algún elemento de la secuencia se repite el elemento no se agregara}

**Otras operaciones**

agregar(conjunto<a>,elemento)

eliminar (conjuntor<a>,elemento)

**Axiomas**

agregar(conjunto<a>,elemento) :se agregar el elemento que queramos a ingresar en el conjunto,en el caso de que este el elemento no se va a ingresar

eliminar (Conjunto<a>,elemento) :se eliminará el elemento del conjunto que indiquemos,(pos:en el caso de que este vacio, nos devolvera que esta vacio )

vacio → Conjuntor<a> {Post; Esta vacio}:se creera el cojunto vacío, sin ningún elemento

a\_partir\_de(Secuencia<a>) → Conjunto<a>:se creerá un conjunto a partir de la secuencia que pasemos como parámetro

{Post:El conjunto contiene los elementos de la secuencia ingresada}

tamaño(cojunto<a>):devolverá la cantidad de elementos que tiene el conjunto

ver(Conjunto<a>) :nos mostrará los elementos que tiene el conjunto

**Exporta**

Conjunto<a>,agregar,eliminar,tamaño,ver,obtener,bool,

TAD ListaEnlazadaSimple<a>

**Igualdad Observacional**

Si a y b son dos Lista Enlazada, a es igual a b si se cumple una de las siguentes condiciones:

-a es vacia y b es vacia

-tienen los mismos elementos, ordenado en el mismo orden

**Usa**

Natural,Bool,Secunecia<a>,String

**Parametro formal**

a

**Observadores básicos**

primero(ListaEnlazada<a>)

ultimo(LIstaEnlazad<a>)

tamaño(LIstaEnlazada<a>)

ver(LIstaEnlazada<a>)

esta(LIstaEnlazada<a>,elemento) Pos{devuelve un booleano}

**Generadores**

vacío → LIstaEnlazada<a> {Post; Esta vacio}

a\_partir\_de(Secuencia<a>) → Vector<a>

{Post:La LIstaEnlazada contiene los elementos de la secuencia ingresada,si algún elemento de la secuencia se repite el elemento no se agregara}

**Otras operaciones**

agregar(LIstaEnlazada<a>,elemento)

eliminar (LIstaEnlazada<a>,elemento)pos {en el caso que no tenga ningún elemento avisaremos que no hay ningún elemento, y si,no esta, devolveremos que no esta el elemmto en la lista}

egregarposicion(LIstaEnlazada<a>,posicion)

**Axiomas**

agregar(LIstaEnlazada<a>,elemento)

eliminar (LIstaEnlazada<a>,elemento)pos {en el caso que no tenga ningún elemento avisaremos que no hay ningún elemento, y si,no esta, devolveremos que no esta el elemmto en la lista}

egregarposicion(LIstaEnlazada<a>,posicion)

vacío → LIstaEnlazada<a> {Post; Esta vacio}

a\_partir\_de(Secuencia<a>) → Vector<a>

{Post:La LIstaEnlazada contiene los elementos de la secuencia ingresada,si algún elemento de la secuencia se repite el elemento no se agregara}

tamaño(LIstaEnlazada<a>)

ver(LIstaEnlazada<a>)

esta(LIstaEnlazada<a>,elemento) Pos{devuelve un booleano}

**Exporta**

LIstaeEnlazada<a>,agregar,eliminar,tamaño,ver,obtener,bool,

**TAD Pila<a>**

**Igualdad Observacional**

Si a y b son dos pilas a es igual a b si se cumple una de las siguientes condiciones:

-a es vacía y b es vacía

-tienen los mismos elementos,apilados en la misma posición.

**Usa**

Natural, Bool, Secuencia<a>

**Parámetro Formal**

a

**Observadores básicos**

tamaño(Pila<a>) → Natural

apilar(Pila<a> , a )→None {Pos: la pila no esta vacía}

tope(Pila<a>)→ a {Pre: la pila tiene al menos un elemento}

**Generadores**

vacia() → Pila<a> {Post: La pila retornada esta vacía}

a\_partir\_de(Secuencia<a>)→ Pila<a>

{Post: La pila contiene apilados los elementos de la secuencia reciba}

otras operaciones

desapilar(Pila<a>)→ Pila<a> {Pre: la pila tiene al menos un elemento}

es\_vacia(Pila<a>) → Bool

**Axiomas**

vacía(): Crea una Pila (sin elementos)

a\_partir\_de(Secuencia<a> s): crea una pila que contienen los elementos de la secuencia s

apilar(Pila<a> , a): Apila en el tope de la Pila<a> el elemento a

tamaño(Pila<a>): Retorna/devuelve la cantidad de elementos de la pila Pila<a>

es\_vacia(Pila<a> ): Retorna/devuelve verdadero si a Pila<a> esta vacía y o n

caso contrario

tope(Pila<a> ): retorna/devuelve el primer elemento de la pila P

desapilar(Pila<a> ): Quita el elemento que se encuentra en el tope de la Pila<a>

Exporta

vacía, a\_partir\_de, tope, tamaño, apilar, desapilar, Natural