|  |  |  |  |
| --- | --- | --- | --- |
|  | **Primer Parcial** - 12/05/22 | |  |
| **Nombre:** | Ignacio Ayerbe |
| **Matrícula:** |  |
| **Mail:** | Nachoayerbe2003@gmail.com |

*Requisito de aprobación del examen: Obtener un mínimo de 5 puntos.*

1. Dadas dos listas ordenadas de enteros, implementar la operación recursiva **combinarListas** que permita generar una lista ordenada con los números enteros de ambas listas recibidas, pero sin incluir aquellos números que existen en ambas listas. *(2 pts)*

Ejemplos:

combinarListas([1,2,3,4], [5,6,7]) = [1, 2, 3, 4, 5, 6, 7]

combinarListas([1,3,4,6,10], [5,6,7]) = [1, 3, 4, 5, 7, 10]

combinarListas([2,5], [1,2,3,4,5,7]) = [1, 3, 4, 7]

proced combinarListas(xs, ys: Lista(entero), ref ts: Lista(entero))

vars:

x, y: entero

si not esListaVacia(xs) entonces

si not esListaVacia(ys)entonces

head(xs, x)

tail(xs)

head(ys, y)

tail(ys)

combinarListas2 (xs, ys, ts, x, y)

sino

copiarLista(xs, ts)

fin si

sino

si not esListaVacia(ys) entonces

copiarLista(xs, ts)

sino

listaVacia(ts)

fin si

fin si

fin proced

proced combinarListas2(xs, ys: Lista(entero), ref ts: Lista(entero), x, y:entero)

vars:

si not esListaVacia(xs)entonces

si x < y entonces

head(xs, x)

tail(xs)

combinarListas2(xs, ys, ts, x, y)

insertar(x, ts)

sino

si not esListaVacia(ys)entonces

si y > x entonces

head(ys, y)

tail(ys)

combinarListas2(xs, ys, ts, x, y)

insertar(y, ts)

sino

head(ys, y)

tail(ys)

combinarListas2(xs, ys, ts, x, y)

insertar(x, ts)

fin si

sino

listaVacia(zs)

fin si

fin proced

#copiarLista(xs:Lista(entero), ref ys: Lista(entero))

(copiar la lista ‘xs’ en este caso a la lista ‘zs’)

1. En un centro de investigación botánico se analizan diversas **especies de plantas** y se necesita modelar una estructura que permita clasificarlas a través de su descendencia entre ellas. Cada especie **desciende de una especie ancestra**, exceptuando una **especie Original** que no tiene ascendencia, o bien, no se descubrió aún alguna.   
     
   A su vez, una especie puede tener **varias especies descendientes**. Estas especies de descendencia inmediata mutan de su ancestra luego de varios años y, **cada descendiente puede llevar distintos años en mutar**. Esta información debe estar registrada en la estructura. Por ejemplo, una especie A puede mutar en una especie B en 50.000 años, mientras que también puede aparecer una mutación C (también descendiente inmediata de A) en 100.000 años.  
     
   Cuando una especie muta en una nueva, existen **diferencias en el código genético**. A fines prácticos, se registran estas diferencias con un número entero que refleja el porcentaje de cambio del código. Entonces, si una especie B muta cambiando sólo el 2% de su código respecto a la antecesora A, se almacenará en la estructura ese número 2. Esto infiere también que la especie B comparte el 98% del código de su antecesora A.  
     
   Una especie tiene un **nombre** asociado, el cual la identifica y también sirve para identificar como **categoría a todas las especies descendientes**. Por ejemplo, las especies descendientes de una especie A, tendrán su propio nombre y también se dice que pertenecen al grupo de tipo A. Otra propiedad que se desea registrar es la **cantidad de agua** que necesitan para sobrevivir, que se simplificará usando la medida de litros por año.

Se solicita realizar lo siguiente:

1. Implementar la estructura del **TAD Especie** que represente a la clasificación de plantas propuesta en el enunciado, junto con los TAD que considere necesarios.   
   Agregar las **firmas de las operaciones** de dichos TADs que utilice en los siguientes puntos.  
   Implementar la operación **crearMutación** que, dada una especie existente y toda la información necesaria para registrar una nueva especie, incorpore a la estructura a la nueva especie como descendiente de la existente. *(1 pt)*

TAD Especie

Tipo:

Especies = registro

original: EspecieDes

nombre: cadena

cantAgua: entero

fin reg

EspecieDes = registro

des: Lista(EspecieDes)

mut: Mutación

nombre: cadena

cantAgua: entero

Fin reg

Mutación = registro

añosMut: entero

codigoG: real

fin reg

# proced CrearEspecieOrg( Esp: Especie, nombre: cadena, cantidadAgua: entero)

(crea una epsecie original y deja la lista vacía para poder cargar nuevas especies)

# proced insertarNuevaPlanta (lista: Lista(EspecieDes), planta: EspecieDes, nombre:cadena, cantAgua: entero, años: entero, codigo: entero)

(insertar una nueva planta, en la referencia pasasda en la lista)

#funcion encontrarEspecieOrg(Lista: Lista(EspecieDes): bool

(verdadero si encuentra la epsecie solicitada, falso si no)

Proced crearMutacion ( especie: Especies, nombre: cadena, cantAgua: entero, años: entero, codigo: entero)

Vars:

Encontrado: bool

nombreExi: cadena

encontrado🡨 falso

nombreExi🡨 especie.nombre

CrearMutacion2(especie.original, nombre, cantAgua, años, codigo, nombreExi, encontrado)

Fin proced

Proced CrearMutacion2 (plantas: EspecieDes, nombre: cadena, cantAgua: entero, años: entero, codigo: entero, nombreExi: cadena, encontrado: bool)

Vars:

desPL : Lista(EspecieDes)

subDes, plantaN: EspecieDes

si no encontrado entonces

desPL 🡨plantas.des

mientras no esListaVacia(desPL) y no encontrado hacer

head(desPL, subDes)

tail(desPL)

si subDes.nombre = nombreExi entonces

encontrado🡨 verdadero

CrearMutaciones2(subDes, nombre, cantAgua, años, codigo, nombreExi)

Fin si

Fin mientras

Sino

insertarNuevaPlanta (desPL, plantaN, nombre, cantAgua, años, codigo)

fin si

fin proced

1. Implementar la operación **maxMutaciones** que, dada una especie Original, devuelva los nombres de las especies con mayor cantidad de descendientes inmediatas (mutaciones directas). El resultado puede ser ninguna (si la especie Original nunca mutó), una o varias especies que comparten la misma cantidad máxima. *(2 pts)*

Proced maxMutaciones(especie: Especies, ref nombres: Lista(cadena))

Vars:

cantMax: entero

error, encontrado: bool

nombre: cadena

nombre🡨 especie. nombre

cantMax🡨 0

error🡨 falso

encontrado🡨 falso

determinarCantMax(especie.original, cantMax, error, nombre, encontrado)

si no error entonces

listaVacia(nombres)

maxMutaciones2(especie.original, cantMax, nombres, nombre, encontrado)

fin si

fin proced

proced determinarCantMax (especie: EspecieDes, ref cantMax: entero, ref error: bool, nombre: cadena, encontrado: bool)

vars:

subPlanta: EspecieDes

plantas: Lista(EspecieDes)

plantas🡨 especie.des

mientras no esListaVacia(plantas) hacer

head(plantas, subPlanta)

tail(plantas)

si subPlanta.nombre = nombre or encontrado entonces

cantMax🡨 cantMax + 1

determinarCantMax(subPlanta, cantMax, error, nombre)

encontrado🡨 verdadero

fin si

fin mientras

si no encontrado entonces

error🡨 verdadero

fin si

fin proced

Proced maxMutaciones2(especie: EspecieDes, cantMax: entero,ref nombres: Lista(cadena), nombre: cadena, encontrado: bool)

Vars:

subPlanta: EspecieDes

plantas: Lista(EspecieDes)

error: bool

cont: entero

plantas🡨 especie. Des

mientras no esListaVacia(plantas) hacer

head(plantas, subPlanta)

tail(plantas)

si subPlanta.nombre = nombre or encontrado entonces

cont🡨 determinarCantMax(especie.original, cantMax, error, nombre, encontrado)

si cont = cantMax entonces

insertar(nombres, subPlanta. Nombre)

fin si

encontrado🡨 verdadero

fin si

fin mientras

fin proced

1. Implementar la operación **mejorResistencia** que, dada una especie Original, devuelva aquellas especies que son más resistentes a la sequía que su especie ancestra inmediata. Diremos que una especie es más resistente si necesita menos agua para sobrevivir. *(2 pts)*

proced mejorResistencia (especie: Especies ,ref nombres: Lista(cadena))

vars:

plantas: lista(EspecieDes)

cantAgua: entero

nombre: cadena

encontrado:bool

encontrado🡨 falso

nombre🡨 especie.nombre

cantAgua🡨 especie.cantAgua

encontrado🡨 falso

si encontrarEspecieOrg (especie.original, plantas) entonces

listaVacia(nombres)

mejorRestistencia2 (plantas, cantAgua, nombres, encontrado)  
 fin si

fin proced

proced mejorResistencia2 (plantas: Lista(EspecieDes), cantAgua: entero, nombres: Lista(cadena), encontrado: bool)

vars:

subPlanta: EspecieDes

head(plantas, subPlanta)

tail(subPlanta)

si no esListaVacia(plantas) entonces

si subPlanta.nombre = nombre o encontrado entonces

si cantAgua > subPlanta.cantAgua entonces

insertar(nombres, subPlanta.nombre)

fin si

encontrado🡨 verdadero

fin si

fin proced

1. Implementar la operación **mutaciónMasLarga** que, dada una especie Original, devuelva la especie que tardó más años en mutar desde la especie Original. A su vez, debe entregar cuál es el porcentaje de código genético que comparte la especie encontrada con la Original. *(3 pts)  
   Nota: Para este ejercicio asumiremos la precondición que existe al menos una mutación desde la Original y que no hay chances que más de una especie haya tardado la misma cantidad de años en mutar desde la Original, por lo cual se devuelve siempre una sola.*

(b\* (a+b))/100

Proced mutacionMasLarga (especie: Especies, ref lista: Lista(Mutacion))

Vars:

mutación: entero

codigoOrg, codigo: real

nombre:cadena

encontrado: bool

xs:lista(Mutacion)

encontrado🡨 falso

nombre🡨 especie.nombre

mutación🡨 especie.añosMut

codigoOrg🡨especie.codigoG

mutacionMasLarga2(especie.original, lista, mutación, codigo, nombre, encontrado)

si encontrado entonces

listaVacia(lista)

listaVacia(xs)

xs^. añosMut🡨 mutacion

xs^. codigoG🡨 (codigo \* (codigoOrg+ codigo)) div 100

insertar(lista, xs)

destruirLista(xs)

fin proced

proced mutacionMasLarga2 (especie: EspeciesDes, ref mutación: entero, ref codigo: real, nombre: cadena, ref encontrado: bool)

vars:

plantas: Lista(EspeciesDes)

subPlantas: EspeciesDes

plantas🡨 especie. Des

mientras no esListaVacia(plantas) hacer

head(plantas, subPlantas)

tail(plantas)

si subPlantas.nombre= nombre o encontrado entonces

si subPlanta.añosMut > mutación entonces

mutación🡨 subPlanta.añosMut

codigo🡨 subPlanta.codigoG

fin si

encontrado🡨 verdadero

fin si

mutacionMasLarga2(especie.original, lista, mutación, codigo, nombre)

fin mientras

fin proced