

Apellido y nombre: _____ Legajo: _____ Curso con Prof: _____

Cantidad de hojas entregadas: _____ Nota: _____ Evaluó Prof: _____

1. Merge (Apareo) de listas enlazadas

Temas evaluados: Listas enlazadas, abstracción procedural, estructuras de control, y lenguaje de programación.

Sea *Merge* una operación definida para listas enlazadas del tipo genérico *T* que recibe dos listas, *a* y *b*, y genera una tercera, *c*, con el apareo o fusión de *a* y *b*.

1a. Escriba en Pascal, C o C++, el encabezado de la función o procedimiento *Merge*.

1b. Diseñe un algoritmo congruente con 1a. El algoritmo puede estar representado por un diagrama estructurado o por código en Pascal C o C++.

*Asuma que las listas *a* y *b* están ordenadas y no contienen repeticiones.*

Respete las siguientes restricciones:

- Invoque a function **ESMenor**(*x*: *T*; *y*: *T*): **Boolean** para comparar los elementos del tipo *T*, retorna verdadero si *x* es menor a *y*, si no, falso. No invoque otras funciones.
- Las listas *a* y *b* no deben ser modificadas.
- La lista *c* debe generarse ordenada.
- Minimice la cantidad de veces que se recorra cualquiera de las tres listas.

2. Suma de Números Astronómicos

Temas evaluados: Pilas, abstracción procedural, estructuras de control, y lenguaje de programación, reusabilidad.

Introducción: En cierto campo de la astronomía los números enteros pueden tener magnitudes muy grandes, y su representación requiere una secuencia extensa de dígitos decimales (0 a 9). Esto imposibilita utilizar los tipos de datos generalmente provistos por los lenguajes de programación. Un número astronómico se almacena en un archivo con una representación especial, y la función *LeerNumero* es capaz de interpretar esa representación. La función *LeerNumero(nombre)* lee la secuencia de dígitos que se encuentra el archivo llamado *nombre* y lo retorna en una variable de tipo de dato *NumeroAstronomico*. *NumeroAstronomico* es una pila de bytes, donde cada byte es un dígito del número astronómico, el dígito menos significativo (el último de la cifra) queda en la cima.

Problema: Se reciben dos archivos, *n1.dat* y *n2.dat*, cada uno contiene un número astronómico, y se requiere informar la sumatoria de ambos.

2a. Implemente la función *SumarNumeros(a,b)* que suma los números astronómicos *a* y *b* y retorna el resultado en una pila de dígitos. En la cima de la pila resultante queda el dígito más significativo de la suma, a la inversa de lo que ocurre con el *NumeroAstronomico*. Las pilas *a* y *b* quedan vacías. Debe utilizar las funciones de la biblioteca al dorso. La implementación de la función puede estar representada en un diagrama estructurado o codificada en Pascal, C o C++.

function **SumarNumeros**(var *a*: *NumeroAstronomico*; var *b*: *NumeroAstronomico*): *PilaDeDigitos*

2b. Implemente un programa que resuelva el problema. Debe invocar a *LeerNumero*, *SumarNumeros*, y a *MostrarNumero*. La implementación del programa puede estar representada en un diagrama estructurado o codificada en Pascal C o C++.


Funciones especiales


function **LeerNumero**(*nombre*: *String*): *NumeroAstronomico*

procedure **MostrarNumero**(var *p*: *PilaDeDigitos*)

Ejemplo número astronómico

N1.dat 9|9|8|6|EOF N2.dat 3|5|EOF
Resultado luego de aplicar *a* = *LeerNumero*(N1) y *b* = *LeerNumero*(N2)

NumeroAstronomico *a* → 

NumeroAstronomico *b* → 

Resultado luego de aplicar *c* = *SumarNumero*(*a*,*b*)

PilaDeDigitos *c* → 

La función *mostrar numero* recibe la pila de dígitos *c* y muestra 10021 como resultado de sumar 9986 + 35.

Observe que los *NumeroAstronomicos* tienen en el tope el dígito menos significativo (por donde debe comenzar la suma) y la *PilaDeDigitos* con la suma tienen en el tope el dígito más significativo.

3. Parámetros

Explique si el pasaje de argumentos por referencia (también llamado por variable) es aplicable para parámetros *in*, *out* ó *inout*.

Apellido y nombre: _____ Legajo: _____ Curso con Prof: _____

Cantidad de hojas entregadas: _____ Nota: _____ Evaluó Prof: _____

Para la resolución del examen, los siguientes procedimientos y funciones *pueden* ser invocados sin ser implementados.
 Por cada procedimiento o función que *decida* invocar **debe reescribir su prototipo** reemplazando correctamente las palabras subrayadas y en cursiva (llamadas variables o no terminales); **la invocación debe ser precisa y concordar con el prototipo**.

Biblioteca genérica de plantillas

Operaciones sobre Archivos

function LeerEspecial(var archivo: *TipoArchivo*; var registro: *TipoRegistro*): Boolean
 Lee un registro desde *archivo* y almacena en *registro*. Retorna pudo leer? True : False.

Estructuras Enlazadas

Operaciones sobre Pilas

procedure Push(var pila: *TipoPuntero*; valor: *TipoInfo*)
 Agrega *valor* a la cima de *pila*.

procedure Pop(var pila: *TipoPuntero*; var valor: *TipoInfo*)
 Saca la cima de *pila* y la almacena en *valor*.

function EstaVacía(pila: *TipoPuntero*): Boolean
 Retorna True si la pila está vacía, si no, False.

Operaciones sobre Colas

procedure Agregar(var frente, fin: *TipoPuntero*; valor: *TipoInfo*)
 Agrega *valor* al final de una cola.

procedure Suprimir(var frente, fin: *TipoPuntero*; var valor: *TipoInfo*)
 Saca de el primer elemento una cola y lo almacena en *valor*.

function EstaVacía(frente, fin: *TipoPuntero*): Boolean
 Retorna True si la cola está vacía, si no, False.

Operaciones sobre Listas

procedure InsertarNodo(var lista: *TipoPuntero*; valor: *TipoInfo*)

procedure InsertarNodoDec(var lista: *TipoPuntero*; valor: *TipoInfo*)
 Inserta en una lista ordenada en forma creciente (decreciente) por el primer campo *valor*.

procedure SuprimirNodo(var lista: *TipoPuntero*; valor: *TipoInfo*)
 Busca un nodo con las características de *valor*, si lo encuentra lo elimina

procedure BuscarOInsertar(var lista, ptr: *TipoPuntero*; valor: *TipoInfo*)
 Busca un nodo con los datos de *valor*, si no lo encuentra, lo inserta y retorna en *ptr* la dirección de memoria creada. Si estaba retorna en *ptr* esa dirección.

procedure InsertarPrimero(var lista: *TipoPuntero*; valor: *TipoInfo*)

procedure InsertarAntesDe(var lista: *TipoPuntero*; valor: *TipoInfo*)

procedure InsertarDespuesDe(var lista: *TipoPuntero*; valor: *TipoInfo*)

procedure InsertarAlFinal(var lista: *TipoPuntero*; valor: *TipoInfo*)

Cada procedimiento crea un nodo con *valor* y lo enlazan relativo a *lista* en la posición indicada por su nombre.

function BuscarPorCampo(lista: *TipoPuntero*; valor: *Tipo*): *TipoPuntero*
 Busca el primer nodo con campo igual a *valor*. Retorna lo encontró? Puntero al nodo : Nil