**Departamento de Desarrollo Productivo y Tecnológico**

**Licenciatura en Sistemas**

***Algoritmos y estructuras de datos - Guia de ejercicios***

**Docentes:** Diego Cañete, Nicolás Perez y Javier Vescio
**Contactos:** diegoleonel.canete@gmail.com, nperez_dcao_smn@outlook.com, javiervescio@gmail.com
**Slack**: https://join.slack.com/t/unlaalgoritmo-pri7290/shared_invite/zt-15ujlnokj-hOm1aJgo~bekbONrVuBACw
**Material digital:** https://1drv.ms/u/s!AgB0dw0E7wKanu43bsncVEEizPhHXg?e=UMH0kQ
**Lista de reproducción:** https://www.youtube.com/playlist?list=PLbwWly6AYXqchZQfRVJrsby-rjWSu-Ys9

# Índice

# Unidad “0” - Nivelación, recordando C

1- Crear un programa que te permita jugar a la Quiniela Plus. ¿Qué es eso?, es un juego de azar donde el jugador compra un **cartón de 8 números del 0 al 99, sin repetir, y ordenados.** En la jugada salen **20 números del 00 al 99, pueden ser repetidos,** en caso de tener **8 aciertos** se gana 11 millones de pesos, con **7 aciertos** 20 mil pesos, con **6 aciertos** 500 pesos y con **5 aciertos** se gana 50 pesos. El cartón tiene la **fecha de emisión**, la **fecha de juego (un domingo)*** y la **dirección de la agencia** que vendió el cartón, el mismo tiene un valor de $70 y además para nutrir el ejercicio pedimos que el cartón este asociado a un **jugador**, con **nombre y dni**.

El programa debe contener como mínimo:

- Comprar cartón automático
- Comprar carton seleccionando los 8 números
- Jugar, creación de los 20 números del partido
- Verificar cartón, saber si has ganado y cuánto has ganado
- Imprimir el cartón, marcando de alguna forma los números que salieron, también guardarlo en un archivo.
- Generar un bucle que te diga cuantas veces tenes que jugar con el mismo cartón para llegar a ganar con 8 aciertos

***Observación: Si hace mucho que no programas con C y estas necesitando un repaso podes visualizar este video***: **Arrancando con C**

**Observación 2: Si estás teniendo problemas para comparar dos array de distintas estructuras podes mirar este video donde les recordamos cómo hacerlo:**
**Comparar Arrays**

*Si pudiste hacer todo lo anterior y te sobró tiempo y curiosidad, vas a poder resolver sin problema el apartado de la fecha y del domingo siguiente al cartón, para eso te invitamos a ver este ejemplo usando fechas de la librería time.h:
**Manejo de fechas, video externo**

**Luego, antes de iniciar la unidad 0, te aconsejamos ver estos videos sobre aspectos básicos de compilación y funcionamiento de un código fuente, al igual que de su separación modular:**

**Compilación y punteros**

**Compilación y módulos en c**

2- **Generar una librería de funciones para vectores de enteros.** La misma tiene que permitir, cargar un vector de forma manual, cargarlo de forma aleatoria, mostrarlo, mostrarlo en forma invertida, contar la cantidad de pares, calcular su módulo (Pitagoras), buscar maximo, buscar mínimo ( y sus posiciones), ordenar el array (por burbujeo, selección e inserción), sumar dos vectores, realizar la multiplicación escalar de dos vectores, realizar

una búsqueda secuencial del vector, realizar una búsqueda binaria sobre el vector. ¿Qué ocurre si les pedimos esto mismo para float o doubles?.

3- **Realizar una librería para poder manipular matrices de enteros**. En esta librería se tiene que tener como mínimo, mostrar la matriz, cargar la matriz manualmente y aleatoriamente, buscar máximo y mínimo y sus posiciones, calcular el promedio por filas y columnas, calcular la suma por sumas y columnas, calcular el promedio general, realizar la suma de matrices y el producto de matrices (con las validaciones necesarias sobre las dimensiones de las mismas).

4- Para agilizar los puntos 2 y 3, ¿tuvo que generar alguna librería/s extra? ¿Cuál/es? ¿En qué momento cree que fue necesario el uso de punteros?

# Unidad “1” - Teoria de datos Abstracto (TDA)

**Si estas con dudas o flojo con el tema de TDA te invitamos a que veas este breve repaso de TDA práctico: Ver video TDA**

1- Construir un TDA completo que permita administrar Dispositivos Tecnológicos (tipo, marca, precio, memoria, etc).

2- Crear un TDA completo para trabajar con una Persona. Una vez creado relacionar a la persona con varios dispositivos tecnológicos.

3- Crear un TDA completo para administrar fechas. Una vez terminado hacer que la persona del ejercicio 2 tenga una fecha de nacimiento.

4- Crear un TDA completo que permita trabajar con Empleados, donde los empleados tienen las mismas variables internas de la Persona del ejercicio 2, pero también tienen un legajo, un sueldo y una empresa. ¿Cómo generaría el TDA para reutilizar todo lo anterior?.

5- Crear un TDA completo que permita trabajar con números complejos. Se tiene que poder implementar la suma, resta, multiplicación, división, módulo y conjugado de un número complejo. No puede faltar el mostrarComplejo.

- Adición: $(a+bi)+(c+di)=(\mathbf{a+c})+(\mathbf{b+d})\mathbf{i}$
- Sustracción: $(a+bi)-(c+di)=(\mathbf{a-c})+(\mathbf{b-d})\mathbf{i}$
- Multiplicación: $(a+bi)(c+di)=ac+adi+bci+bdi^2=(\mathbf{ac-bd})+(\mathbf{ad+bc})\mathbf{i}$
- División:

$$\frac{a+bi}{c+di}=\frac{(a+bi)(c-di)}{(c+di)(c-di)}=\frac{ac-adi+bci-bdi^2}{c^2-d^2i^2}=\frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

**Nota:** Se puede reemplazar $i^2$ por $-1$.

6 - Ejercicio cuatrimestral, para irlo mejorando a medida que vemos más temas. Crear un programa que permita generar facturas para ventas de un comercio.

- El comercio vende **Productos** (alimenticios), de los que queremos saber **nombre**, **precio**, **código** y **cantidad disponible**.

- Al realizar una compra tenemos que facturar, de la **Factura** nos interesa saber su **número**, la **fecha**, los datos del **Cliente** (**dni, nombre**) y el **Detalle** de la compra ( **nroDetalle**, **producto**, **cantidad**, **precio** y **precio total**). Una vez generados todos los detalles la Factura tiene que tener un **precio final**.

- Una factura tiene **muchos detalles**, proponemos solucionar ese problema con un **array**, en la próxima unidad lo haremos mucho mejor.

- Generar un **mostrarFactura** donde se vean todos los datos impactados en la compra.

# Unidad “2” - Listas

1 - Analizar el ejemplo básico de listas enlazadas (int) visto en clase, una vez analizado, agregarle pre y post condiciones a todo el programa y comentar todas las líneas para comprender su funcionamiento.

Video introductorio listas
Ejemplo básico - Lista enlazada (int)

*2 - Tomando como base lo aprendido en el item anterior agregar primitivas a la librería de listas, para que se pueda ordenar la lista y buscar elementos en la lista. (Siempre con int).*

Ayuda para ordenamiento
Ayuda para busqueda

3 - Modificar la librería anterior ( o crear una nueva) para poder enlistar float.

4 - Como se tuvo que haber entendido en el punto 3, no tiene sentido generar una lista para cada tipo de datos, para esto se puede generalizar la lista haciendo que el dato del nodo sea un dato cualquiera. Analizar el ejemplo dado a continuación donde se muestra como generalizar una lista para cualquier dato. Hacer funcionar la lista para algún otro dato personalizado.

Ejemplo lista enlazada (Dato)

5 - Reconstruir los TDA de la Unidad 0, para que la Persona tenga una lista de dispositivos tecnológicos, esta persona puede comprar dispositivos, vender dispositivos, buscarlos y mostrarlos en todo momento. **Es importante el correcto uso del encapsulamiento, ya que la lista está dentro de una estructura. Recordar los getters de programación.**
6 - **Construir los TDA que sean necesarios para que un determinado equipo tenga una lista de jugadores**, y a su vez esos jugadores tengan una lista de títulos en su historial.

7 - Analizar la lista circular, ¿cuáles son las ventajas y desventajas sobre la lista simplemente encadenada?
**Para profundizar sobre la teoría y estructura de los distintos tipos de listas, les dejamos estos dos links externos:**
Tipos de listas - Wiki
Tipos de listas - ALBERTO CARDENAS

8 - Crear una lista circular de un TDA de Persona. Pueden apoyarse en el siguiente código fuente o modificar las listas que crearon en ejercicios previos.
Ejemplo lista circular (Dato) - Código externo

9 - Crear un programa que permita tener una lista de alumnos, con nombre, dni y promedio. Se pide que luego de tener por lo menos 5 alumnos se ordene la lista por algún campo, PERO que no se pierda la lista original, es decir crear un duplicado ordenado. Una vez terminado este ejercicio modificar lo que sea necesario para realizar lo mismo pero con una lista de Inmuebles, donde cada inmueble tiene un valor, año de construcción, dirección y un dueño. Se recomienda utilizar listas void, para modificar lo menos posible.
Ejemplo sencillo lista void

10 - Crear un programa que permita enlistar a las Personas que asistieron en un Estadio de fútbol, además cada Persona tiene una lista de Motivos por los que asistir al estadio. (2 o 3 personas con 2 o 3 motivos, lo importante son las dos listas con Void).

11 - ¿En qué casos cree conveniente una lista doblemente enlazada sobre una lista simplemente enlazada?. Generar una lista doblemente enlazada que permita trabajar con un TDA de Edificios.
Ejemplo lista doblemente enlazada

12 - Modificar el ejercicio 6 de la Unidad “0” (Factura), eliminando los array por listas.

# Unidad “3” - Pilas y Colas

1- Analizando los ejemplos de Colas, realizar las primitivas de Pilas.

2- Crear una pila de enteros, e irlos desapilando y mostrando.

3- Crear una pila de Monedas (valor, color, año), e irlas sacando una por una, las monedas tienen un valor y un año.

4- Modificar el ejercicio de Facturación de forma que cada vez que se haga una factura la misma se archive en una pila o cola de facturas.
5- Crear un programa que permita generar una Cola de Estudiantes para ingresar al sistema Guarani, la cola no puede superar los 5 estudiantes. Es importante el correcto uso del encapsulamiento, ya que la lista está dentro de una estructura. Recordar los getters de programación.

# Unidad “4” - Árboles

Resumen y práctica de Arboles binarios

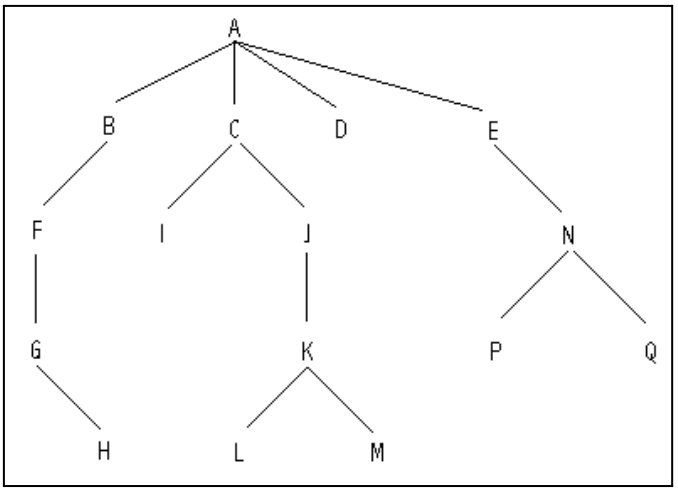
1 - Construir un árbol cualquiera, donde en cada nodo se guarde un entero aleatorio cualquiera, se irán insertando en el árbol nodos con números aleatorios, el primer nodo será la raíz.  Hacerlo con por lo menos 10 números. Mostrar cómo quedó el árbol (de la manera que más fácil le resulte).

2 - Del árbol anterior, o de cualquier otro mostrar cómo quedarían los 3 recorridos (IN-PRE-POST/ORDEN). Explicar en qué casos se usa cada uno.

3 - Crear una función para calcular la altura del árbol del ejercicio anterior.

4 - Responder las siguientes preguntas sobre el árbol ilustrado:

¿Qué nodo es la raíz?
¿Cuántos caminos diferentes de longitud tres hay?
¿Es un camino la sucesión de nodos HGFBACI?
¿Qué nodos son los ancestros de K?
¿Qué nodos son los ancestros propios de N?
¿Qué nodos son los descendientes propios de M?
¿Qué nodos son las hojas?
¿Cuál es la altura del nodo C?
¿Cuál es la altura del árbol?
¿Cuál es la profundidad del nodo C?
¿Cuál es el hermano a la derecha de D?
¿Es I hermano a la derecha de F?
¿Está F a la izquierda de J?z
¿Está L a la derecha de J?
¿Qué nodos están a la izquierda y a la derecha de J?
¿Cuántos hijos tiene A?
Listar los nodos del árbol en preorden,postorden e inorden.



5 - El recorrido en preorden de un determinado árbol binario es: GEAIBMCLDFKJH y en inorden IABEGLDCFMKHJ. Resolver:

A) Dibujar el árbol binario.

B) Dar el recorrido en postorden.

C) Diseñar una función para dar el recorrido en postorden dado el recorrido en preorden e inorden y escribir un programa para comprobar el resultado del apartado anterior.

6 - Realizar un algoritmo que permita buscar un elemento en un árbol.

Ayuda para práctica de árboles AVL

7- Realizar un algoritmo que calcule la altura de un árbol binario (agregarla a los ejemplos dados en la clase). Además un algoritmo que calcule la profundidad de cualquier nodo. ¿Se puede reutilizar lo anterior?

8 - Dada la secuencia de claves enteras: 100,29,71,82,48,39,101,22,46, 17,3,20,25,10. Representar gráficamente el árbol AVL correspondiente.

Link para verificar resultados sobre insertar AVL

9 - ¿Qué conjeturas puede sacar de estos ejercicios? ¿En qué tipo de árbol se genera el árbol de altura mínima? ¿Cuáles serían las ventajas y desventajas de uno y otro esquema de representación de árbol?.
Todo lo hasta ahora visto respecto a árboles AVL está relacionado con el rebalanceo después de la inserción de algún nodo que rompa el balance del árbol AVL. Sin embargo ¿qué pasa con la eliminación de elementos?

10 - Con el ejercicio de Facturas, generar 10 facturas e irlas acomodando en un árbol considerando su valor final. ¿Qué se ganaría haciendo esto? ¿Qué tanto sentido le encuentran a hacer esto? ¿En qué casos cree que valdría la pena almacenar de esta forma información de facturación o algo similar?

Ejemplos de Arboles Binarios y AVL en C

# Unidad “5” - Grafos

Antes de iniciar aconsejamos afianzar las definiciones de grafos:

PPT de grafos

1 - Comentar las líneas de código del ejemplo práctico de grafos. Entender la lógica de los algoritmos BFS y DFS. Una vez entendido los algoritmos, crear un ejemplo parecido pero que reutilice las listas y colas vistas en las unidades anteriores para mejorar el programa. Notar que en el ejemplo la cola y la lista es un array.

Ejemplo práctico de Grafos - C

2 - Sobre el ejemplo base desarrollado en C (ejercicio anterior), crear un procedimiento que imprima la matriz de adyacencia.

3 - ¿Qué es el recorrido DFS? Realizar el recorrido DFS del grafo de la imagen I y partiendo de 3 vértices distintos.

4 - ¿Qué es el recorrido BFS? Realizar el recorrido BFS del grafo de la imagen II y partiendo de 3 nodos distintos.

5 - Realizar la lista de adyacencia y la matriz de adyacencia de los grafos dados en las imágenes III y IV.

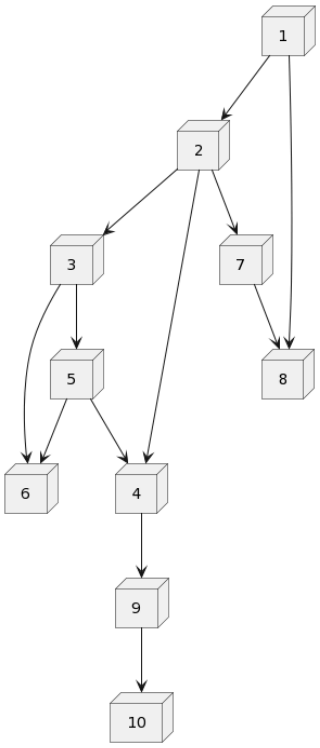



**Imagen I**

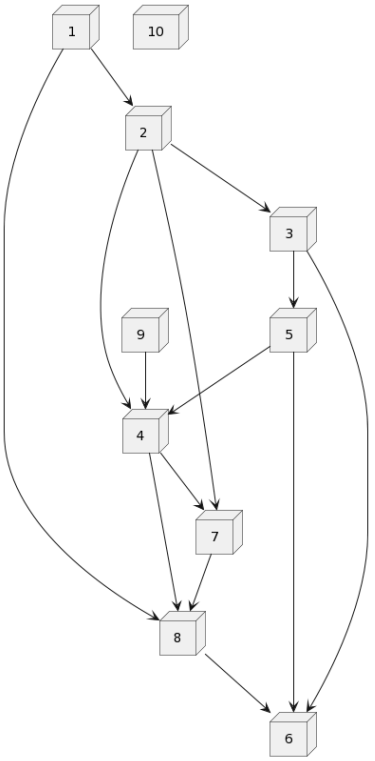



**Imagen 2**

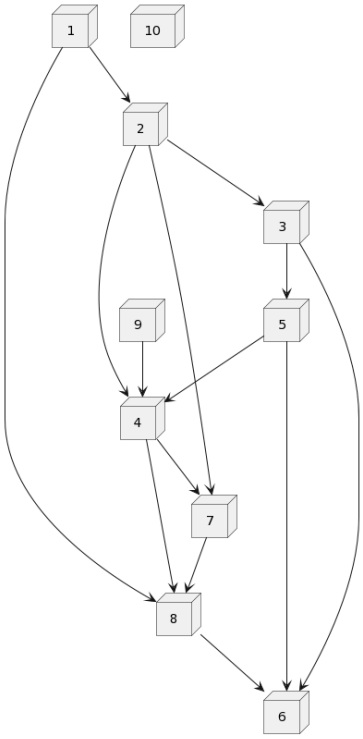



**Imagen III**

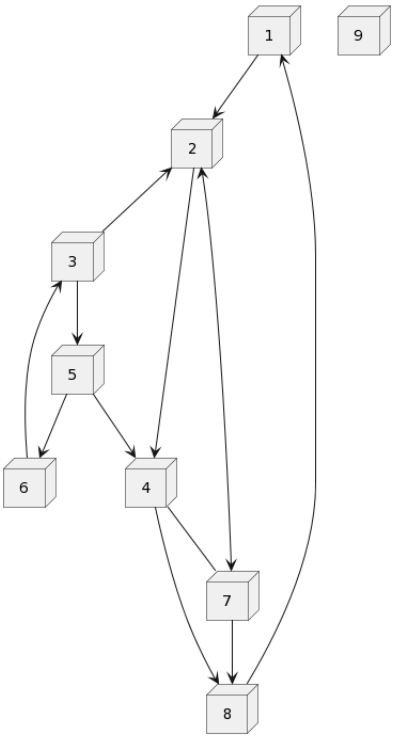



**Imagen IV**

# Unidad “6” - Eficiencia de Algoritmos

# Enunciado TP

## Enunciado 2023 - C2

- Continuar el enunciado del 1er parcial, estará disponible luego del parcial en el material digital.
- Sobre dicho enunciado incorporar un árbol binario del **tipo void**, en el apartado que le sea más apropiado.
- Crear un menú interactivo para trabajar con todas las estructuras
- Entregar un video de menos de 10 minutos donde se lo vea en funcionamiento
- En el video, explicar verbalmente y/o gráficamente dónde podría usar el concepto de grafos en su tp.

Archivos del parcial

## Enunciado 2023 - C1

- Crear un menú que permita operar con el programa.

- El programa deberá administrar como mínimo 3 estructuras propias, bajo el paradigma de TDA, el programa deberá utilizar por lo menos una pila void y una cola void.
- El ejercicio es libre, a modo de ejemplo, Un Sistema de tickets, tiene una cola de Personas y una pila de actividades.
- Con el menú si o si se deberá poder apilar y desapilar, encolar y desencolar y acceder a la información que corresponda de la pila y la cola.
- El TP se deberá subir al campus virtual y adjuntarle un video de menos de 5 minutos donde se lo ve en funcionamiento.

## Enunciado 2022

- Se deberá entregar todo el código fuente comprimido en rar en el campus virtual antes de la fecha indicada.
- Además se deberá adjuntar un video de menos de 10 minutos mostrando el funcionamiento del programa y comentando los bloques importantes de la programación.
- La nota del TP tiene el mismo peso que el parcial, solo pueden entregar el TP los que aprobaron el primer parcial o su recuperatorio.
- ***Sobre el ejercicio:*** Realizar un programa bajo el paradigma de TDA, que relacione un **mínimo de 3 estructuras** (sin contar las estructuras propias de listas, pilas y colas), en dicho programa tiene que existir un **menú** que permite ver en acción una **PILA** de algunas de las estructuras anidadas y además una **COLA** de otras estructuras. Es importante que sean creativos con el programa y que tenga sentido la implementación, por ejemplo **"un negocio tiene una cola de clientes y cada cliente tiene una pila de compras en el carro"**.

- (Anulado 2022) El programa tiene que tener una persistencia **básica**, por lo menos se debe guardar la info en un txt ( si se puede recuperar aún mejor), *no es obligatorio pero si en vez de persistir en txt se animan a persistir en mysql les dejo un video de como se haria (si ya cursaron IBD).*

- Se recomienda que una vez que arranquemos a ver Listas, ya tengan en mente el TP y lo hagan poco a poco así lo vamos viendo sobre la cursada, será útil para que entrenen para el parcial.

# Anexo - Ayudas:

**Potencia y raíz en C, librería math.h: Enlace externo**

Ayudas de operaciones (ejercicios 2 - 3, de la Unidad -1)

**Suma de vectores:**

Supongamos que tenemos los vectores $\vec{A}$ = (4, 3) , $\vec{B}$ = (2, 5) .

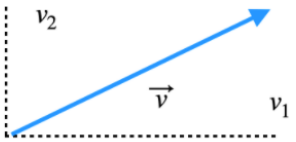
Para conocer el vector suma $A \overrightarrow{+} B$ sólo tenemos que sumar, respectivamente, las componentes X y las componentes Y:

$A \overrightarrow{+} B$ = (4+2, 3+5) = (6, 8)

**Módulo de un vector:**

$$\vec{v}=(v_1, v_2)$$

$$|\vec{v}| = \sqrt{v_1^2 + v_2^2}$$




**Producto escalar entre vectores:**

Dados dos vectores:

$$\vec{a} = (a_1, a_2, a_3)$$

$$\vec{b} = (b_1, b_2, b_3)$$

*Vectores*

El producto escalar se calcula de la siguiente forma:

$$\vec{a} \cdot \vec{b} = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$

*Producto escalar*

**Suma de matrices (Definida sólo si las matrices tienen la misma dimensión):**

$$A=\begin{pmatrix}2 & 0 & 1\\ 3 & 0 & 0\\ 5 & 1 & 1\end{pmatrix} \quad B=\begin{pmatrix}1 & 0 & 1\\ 1 & 2 & 1\\ 1 & 1 & 0\end{pmatrix}$$

$$A+B=\begin{pmatrix}2+1 & 0+0 & 1+1\\ 3+1 & 0+2 & 0+1\\ 5+1 & 1+1 & 1+0\end{pmatrix}=\begin{pmatrix}3 & 0 & 2\\ 4 & 2 & 1\\ 6 & 2 & 1\end{pmatrix}$$

**Producto de matrices (sólo para matrices cuadradas):**

$$\begin{pmatrix}1 & 4 & 7\\ 2 & 5 & 8\\ 3 & 6 & 9\end{pmatrix}\begin{pmatrix}1 & -1 & 2\\ 2 & -1 & 2\\ 3 & -3 & 0\end{pmatrix}=$$

$$=\begin{pmatrix}1+8+21 & -1-4-21 & 2+8+0\\ 2+10+24 & -2-5-24 & 4+10+0\\ 3+12+27 & -3-6-27 & 6+12+0\end{pmatrix}=$$

$$=\begin{pmatrix}30 & -26 & 10\\ 36 & -31 & 14\\ 42 & -36 & 18\end{pmatrix}$$

**Árboles - Ejercicio 5 - C:**

Algoritmo recursivo que soluciona el problema. Pensarlo y comentarlo:

```
void post(char *pre,char *in,char *pos,int n){
  int longIzqda;

  if(n!=0){
    pos[n-1]=pre[0];
    longIzqda=strchr(in,pre[0])-in;
    post (pre+1,in,pos,longIzqda);
    post (pre+1+longIzqda,in+1+longIzqda,pos+longIzqda,n-1-longIzqda);
  }
}
```