# Práctico 7: Ejercicios tipo Examen Parcial y Final

## 1) Parcial 2016

Se tiene un archivo de ***codigo*** (entero) en el que los registros están ordenados de menor a mayor y un árbol también ordenado ascendente con ***codigo*** (entero) que además tiene el atributo ***nivel***(0 para la raíz, 1 para los hijos, etc [indica la profundidad en el árbol]). Solicite al usuario el ingreso de ***codigo_buscado*** (un número entero) y realice lo siguiente:

1) Busque el ***codigo_buscado*** en el archivo y en el árbol, si está en el archivo y NO en el árbol agréguelo en el árbol en el lugar correspondiente (y con el valor de ***nivel*** que le corresponde según su profundidad) de lo contrario no haga nada.
2) Arme una lista simple de nodos con (***codigo*** y ***nivel***) ordenada de mayor a menor por ***nivel*** con todos los nodos del árbol que cumplen las siguientes condiciones:
   a) El ***codigo*** está entre ***codigo_buscado*** y ***codigo_buscado + 100***.
   b) El nodo posee cantidad impar de descendientes (hijos, nietos, etc).

   La lista debe estar en todo momento ordenada por ***nivel***. No importa el orden entre los nodos del mismo ***nivel***.

Notas: 1) No use bisección en el archivo. 2) Para la promoción: No recorrer de más archivo ni árbol. 3) El punto 2 es fundamental para aprobar. 4) Realice el DE y programa en Pascal.

## 2) Recuperatorio 2016

Se tiene una lista apuntada por ***CLIENTES*** cuyos nodos tiene la siguiente información: ***código_cliente***(entero), ***facturas***(puntero a ArbolFact), ***sig_cli*** (puntero a nodo), ***montoAdeudado***(integer, inicialmente en cero), ***sig_deudor***(puntero a nodo).
La lista ***CLIENTES*** está ordenada ascendente por ***código_cliente*** a través del puntero ***sig_cli***. Cada árbol apuntado por facturas corresponde a un árbol binario de facturas impagas de ese cliente, ordenado por nro factura y cuyo nodo contiene además, el ***monto_factura***.
Los punteros ***sig_deudor*** de cada nodo de la lista están ***nil***. A su vez existe otro puntero del mismo tipo ***INICIO_Deudas*** que también está en ***nil***.
Asimismo Se tiene un archivo ARC_CLIENTE cuyos registros contienen un solo campo, ***codigo_cliente***(entero).
**Se pide:** Se creará un un nuevo orden en la lista ***CLIENTES*** a través de los punteros ***sig_deudor*** para todos los nodos que cumplan la siguiente condición:
Leer el archivo ARC_CLIENTE y para cada cliente que está en el archivo se busca en la lista ***CLIENTES***. Si está, se debe actualizar el campo ***montoAdeudado*** con la suma de todas las facturas del árbol y se actualiza el puntero ***sig_deudor*** de este nodo para mantener a la lista ordenada ascendentemente por ***montoAdeudado***.
***INICIO_Deudas*** es el puntero que permite acceder al primer nodo de la lista ***CLIENTE*** ordenada por ***montoAdeudado***.

## 3) PREFINAL 2016

Se tienen cargados en el archivo Equipos.dat los datos referidos a equipos participantes de un torneo deportivo. El archivo no se encuentra ordenado bajo ningún criterio. Para cada equipo se guarda: nombre (de tipo string), puntaje (integer).

Se pide que implemente una función / procedimiento en Pascal que genere un árbol ARB_EQUIPOS en memoria principal con todos los datos del archivo teniendo en cuenta lo siguiente:

· El árbol debe estar ordenado alfabéticamente por nombre del equipo.

- Si ingresa un equipo con un nombre que ya fue incorporado al árbol, no debe incorporarse.
- Los nodos del árbol deben tener un vínculo adicional de tal manera que los equipos puedan ser listados ascendentemente por puntaje mediante ese vínculo. Ese nuevo orden es accedido por un puntero inicial ORDEN_PUNTAJE . **NOTA:** El árbol debe estar vinculado y ordenado correctamente por puntaje en todo momento (ante cada nuevo equipo que se agrega). No puede usar estructuras auxiliares. Debe definir: diagrama de estructura, programa principal, procedimientos y funciones, constantes, tipos y variables.

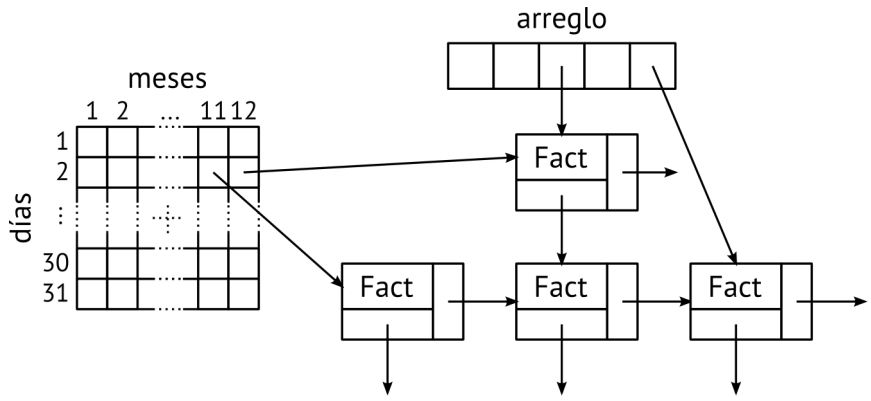
4 ) Se tiene un árbol de números positivos ordenado por los mismos. Se pide que lo recorra in-order llevando en todo momento la suma acumulada de los nodos por los que se pasó. Debe imprimir todos los nodos que cumplan con la condición que la suma acumulada es igual a la suma del subárbol derecho (mayores) del mismo. La función que suma el subárbol derecho no debe seguir recorriendo nodos si detecta que la suma se superó. Realice todas las definiciones, DE y el programa principal. No puede utilizar estructuras auxiliares.

5) Se tiene un arbol CIUDADES donde cada nodo tiene el nombre de la ciudad, la provincia a la que pertenece y la cantidad de habitantes. El árbol está ordenado por nombre de ciudades.
Se pide que arme una lista POBLACION a partir de los nodos del árbol que se encuentran dentro de un rango ciudad1:ciudad2 que debe pedir por teclado. No recorrer el árbol innecesariamente.
La lista población estará formada por un nodo que contiene la provincia y un entero con la cantidad de población acumulada correspondiente a todas las ciudades de esa provincia que caen en el rango.
La lista población debe estar siempre ordenada ascendentemente por la cantidad de población de provincia. No puede usar estructuras auxiliares.

6) Se tiene un árbol binario con información de Exámenes de Alumnos, cada nodo posee número de alumno, materia, notaFinal. El árbol está ordenado por número de Alumno, cada Alumno puede tener varios nodos, uno por materia (las materias para un alumno no se repiten).
Se desea que cree una lista doblemente vinculada con todos los alumnos cuyos números estàn FUERA de un rango NumeroAlumnoMinimo:NumeroAlumnoMaximo, es decir, son menores estrictos a NumeroAlumnoMinimo o mayores estrictos a NumeroAlumnoMaximo.
Cada nodo de la lista poseerá el número de Alumno, y la nota màs alta de final obtenida para todas las materias que rindió. La lista debe estar en todo momento ordenada ascendentemente por la nota más alta.
Debe recorrer una única vez el árbol y no pasar por nodos innecesariamente.

7) Declare la siguiente estructura de datos:

*Una matriz de día por mes donde cada componente es puntero a la lista de los nodos factura correspondientes a esa fecha, y un arreglo de clientes en el cual cada componente contiene el nro de cliente y un puntero a la lista de nodos-factura correspondientes a ese cliente (ordenada por fecha). Cada nodo-factura tiene entonces dos campos puntero y los siguientes campos de datos: nro de factura, fecha, nro de cliente, importe total, y un campo booleano que indique si la factura está paga.*




Codifique:

a) El procedimiento que inicialice la estructura de datos;
b) El procedimiento que permite el ingreso de los datos de una factura y actualice la estructura de datos;
c) La función que dado el nro de cliente devuelva el importe total de las facturas impagas;
d) El procedimiento que dada una fecha emita el listado con el detalle de las facturas correspondientes;
e) El programa principal que invoque los procedimientos / funciones anteriores.

8) Declare la siguiente estructura de datos de un diccionario de sinónimos:

*Un arreglo con un índice de las letras del alfabeto, en el cual cada componente es puntero a la lista ordenada de palabras que comienzan con esa letra, donde además las palabras que son sinónimos están vinculadas por medio de una lista circular.*

Codifique:

f) El procedimiento que inicialice la estructura de datos;
g) El procedimiento que permite el ingreso de dos palabras que son sinónimos (puede ser que ninguna, una o las dos palabras ya existan en el diccionario);
h) El procedimiento que dada una letra del alfabeto emita el listado de todas las palabras que comiencen con esa letra con sus respectivos sinónimos;
i) Un programa principal que invoque los procedimientos anteriores.