# ADP 2013
## Práctica de repaso del segundo semestre

### Ejercicio 1

Se dispone de una lista con la información de archivos almacenados en un "*pen drive*". De cada archivo se conoce: número de archivo, fecha de creación, fecha de última actualización, extensión y tamaño (expresado en KB). Esta estructura no posee orden alguno.

a) En base a la lista que se dispone generar una nueva estructura que contenga dos campos: tamaño (expresado en KB) y una lista con la información de los archivos que poseen dicho tamaño. Esta nueva estructura debe ser eficiente para realizar búsquedas y debe estar ordenada por tamaño.

Una vez generada la estructura del inciso a) se pide:

b) Realizar un módulo que reciba la estructura generada en el punto a), un tamaño y un número de archivo y elimine dicho archivo si existe.

c) Calcular e informar la cantidad de archivos cuyo tamaño es mayor que 1024KB y menor a 102400KB.

d) Calcular e informar la cantidad de archivos que poseen extensión "txt" o "pdf".

### Ejercicio 2

Un supermercado dispone de una lista donde se tiene almacenada la información de las ventas que se realizaron durante el año. De cada venta se conoce: código de producto, cantidad vendida, fecha y número de cliente. Esta estructura esta ordenada por código de producto y puede haber varias ventas que tengan el mismo código de producto.

Además, el supermercado dispone de otra estructura (que permite realizar la búsqueda de manera eficiente) donde almacena para cada código de producto el stock actual y el stock mínimo permitido. Esta última estructura se encuentra ordenada por código de producto.

A partir de la estructuras disponibles se pide:

a) Realizar una actualización del stock actual de cada producto vendido.

b) Generar una nueva estructura donde para cada código de producto vendido aparezca el monto total recaudado (ordenada en forma ascendente por código de producto).

c) Informar el código del producto con mayor stock actual entre aquellos productos cuyo código de producto se encuentre entre 10 y 50.

**Nota**: Modularizar. Definir claramente el programa principal y las estructuras de datos que utilicen. Para los incisos de a) debe recorrer una sola vez la lista disponible.

**Ejercicio 3**

**TAD** Tinvitado;
**Interface**
**Type** exportado invitado;
**Procedure** crearInvitado (**var** i: invitado; nombre: string; apellido: string; distancia: integer; es_familiar: boolean); { *Crea un invitado con la información recibida como parámetro. "Distancia" representa a cuántos kilómetros está el invitado del salón de fiestas y "es_familiar" representa si el invitado es familiar o no de Fernandito.}*
**Function** verDistancia (i: invitado): integer; {*Retorna la distancia en km. al salón de fiestas para el invitado "i"*}
**Function** verFamiliar (i: invitado): boolean; {*Retorna si el invitado "i" es familiar de Fernandito*}
**Procedure** verApellido (i: invitado; **var** apellido: string); *{Devuelve el apellido del invitado "i"}*

Se acerca el cumpleaños de Fernandito y está organizando su fiesta, y para ello dispone de una lista simple con la información de los invitados (tipo exportado del TAD) ordenada por apellido del invitado (*tener en cuenta que puede haber varios invitados con el mismo apellido).*

Realizar un programa que:

A. Genere, a partir de la estructura disponible, una nueva estructura que contenga por cada distancia en km al salón de fiestas, la cantidad de invitados que son familiares de Fernandito. Esta estructura debe estar ordenada por distancia en km y debe ser eficiente para la búsqueda por dicho criterio.

Una vez generada la estructura del punto A, se pide:

B. Realizar un módulo que calcule eficientemente la cantidad total de invitados que son familiares de Fernandito y que se encuentran a menos de 150 km. de distancia del salón de fiestas.
C. Realizar un reporte en pantalla con el siguiente formato

Apellido: **AAA**

Cantidad total de invitados con Apellido AAA: **…**

…

Apellido: **JJJ**

Cantidad total de invitados con Apellido JJJ: **…**

…

**El apellido con mayor cantidad de invitados es:** *DDD*

**Ejercicio 4**

Se dispone de una estructura con la información de los doctores que atienden en una clínica. De cada uno se registra: nombre, especialidad, categoría de IOMA (A,B,C) y listado de turnos disponibles ordenada (el turno es un string con el siguiente formato “AAAA-MM-DD HH-MM”). Esta estructura esta ordenada por nombre de doctor y debe ser eficiente para la búsqueda.

Se pide:

a) Leer la información desde teclado de personas que piden un turno. A cada persona se le solicita: su nombre y el nombre del doctor con el que se quiere atender. La lectura termina con la persona con nombre ZZZ. A cada persona se le debe informar la categoría del doctor y el turno reservado (en caso de existir turno libre se le da el más próximo, caso contario informarle “no hay turno”).

Luego de realizar el punto a)

b) Informar los nombres de los doctores dermatólogos que no poseen más turnos disponibles.
c) Realice un módulo que reciba la estructura principal, el nombre de un doctor y un turno cancelado por un paciente. El módulo debe incluir en la lista de turnos disponibles el turno cancelado.

**Ejercicio 5**

Una fábrica de computadoras requiere un sistema para realizar el armado de PCs, según los pedidos que recibe. El sistema dispone de un catálogo, donde para cada modelo de PC contiene: Código de PC, Nombre de PC, Precio y una lista con los Códigos de todos los Componentes que requiere para su armado. Este catálogo no posee orden alguno.

También, la fábrica dispone de un inventario, donde por cada Código de Componente tiene el stock actual y el stock mínimo. Este inventario se encuentra ordenado por Código de Componente y es eficiente para la búsqueda por dicho criterio.

a) Realice un módulo que reciba el catálogo de PCs, el inventario y un Código de PC, y retorne si finalmente se logró armar la PC cuyo Código se recibió por parámetro. Recordar que para armar una PC se requiere la existencia de stock de todos sus Componentes. En caso de poder realizar el armado de la PC, debe realizar la actualización del stock en el inventario.

b) Realice un módulo que reciba el inventario y retorne una lista con todos aquellos códigos de componentes cuyos stock actual está por debajo del stock mínimo.

Utilizando los módulos anteriores, realice un programa que lea Códigos de PCs hasta llegar el código 0 (cero) el cual no debe procesarse. Una vez finalizada la lectura, informe la cantidad de PCs que fueron armadas y los Códigos de Componentes que necesiten reposición de stock.