PRÁCTICA 6
TAD - Tipo Abstracto de Datos (continuación)

1.- a) Implemente el TAD UrnaElectronica, cuya interface se muestra a continuación. Tener en cuenta al momento de implementar que una UrnaElectronica puede manejar a lo sumo 50 listas.

TAD UrnaElectronica;
interface
type exportado urna;
procedure CrearUrna (var u:urna; nombre: string; cant_listas:integer)
// Crea la urna con nombre "nombre" y los códigos de las agrupaciones enumeradas de 1 a cant_listas y sin votos
function cantidadListasPosibles(u:urna): integer;
// Retorna la cantidad de listas disponibles para votar. Las listas se numeran de 1 a la cantidad
function numeroListaValida (u: urna, num: integer) : boolean;
// Retorna si el número que ingresa como parámetro corresponde a una lista válida
procedure votarPorLaLista(var u: urna; num_lista: integer) ;
// Actualiza la urna sumando 1 a la lista cuyo numero ingresa como parámetro
procedure votarEnBlanco(var u:urna);
// Actualiza la urna sumando 1 al contador de votos en blanco
function cantidadDeVotosEnBlanco(u:urna): integer;
// Retorna la cantidad de votos en blanco que se emitieron
function cantidadDeVotosPorLaLista(u: urna; num_lista: integer);
// Retorna la cantidad de votos que recibió la lista cuyo numero ingresa como parámetro

b) Utilizando el TAD UrnaElectronica resolver el siguiente problema:

La facultad de Informática va a implementar las elecciones por medio de voto electrónico. Para ello dispone de una estructura que representa el padrón de alumnos que cumplen las condiciones necesarias para votar en las elecciones estudiantiles de la facultad. De cada alumno se conoce: Número de Alumno, DNI y Apellido y Nombre. La estructura se encuentra ordena por el DNI del alumno de forma ascendente.
La elección se implementará por medio de una urna electrónica. Dicha urna contabiliza los votos que cada alumno realiza para cada una de las listas que se presentan o si el alumno elige votar en blanco.
Realizar un programa que utilizando el TAD "UrnaElectronica":

I. Cree una Urna con una cantidad de listas que se lee.
II. Simular el proceso de llegada de los alumnos a votar. Cuando el alumno se presenta a votar ingresa su DNI y se debe controlar que el mismo sea válido (esté en el padrón) y que no haya votado aún. En caso de que todo esté correcto, se debe solicitar que el alumno ingrese la lista por la que vota o si vota en blanco. La llegada de alumnos finaliza cuando viene un alumno con DNI "0".
III. Luego de finalizada la votación calcular e informar la lista ganadora.

Nota: Tener en cuenta que se debe registrar en el padrón si el alumno votó o no.

2.- a) Implemente el tipo exportado y las operaciones del siguiente TAD

```
TAD Alumno;
interface
  type exportado alum;
        procedure crearAlumno (var a:alum; nombre:string; dni:string; edad:integer);
        // Crea el alumno a con nombre nombre, dni dni edad edad y con notaDesempeño en 0.
        procedure verNombre(a: alum; var nom:string)
        // Devuelve en nom el nombre del alumno a
        procedure verDni(a: alum; var dni:string)
        // Devuelve en dni el dni del alumno a
        function verNotaDesempeño(a: alum): integer;
        // Devuelve la nota desempeño del alumno a
        procedure modificarDni(var a: alum; dni:string)
        // Modifica el dni del alumno a con el nuevo dni
        procedure modificarNotaDesempeño(var a: alum; nota:integer)
        // Modifica la notaDesempeño del alumno a con la nueva nota.
        procedure asignarAlumno(var a1: alum; a2:alum)
        // Asigna el alumno a2 a el alumno a1.
```

b) Una cátedra de la Facultad de Informática posee una Lista Principal de Alumnos que están cursando en ella (definida utilizando el TAD Alumno) ordenada por DNI del alumno. Además dispone de una Lista de Parciales que contiene los resultados de **cada uno de los parciales del alumno**. De cada parcial se conoce el número del parcial, DNI del alumno y el puntaje obtenido en dicho parcial. Esta información se encuentra ordenada por DNI del alumno. Considerando que se dispone de esta información se pide:

I. Actualizar la información de la notaDesempeño de la Lista Principal de Alumnos para los alumnos que aparecen en la Lista de Parciales, con el promedio de las notas obtenidas en sus parciales.

II. Luego de realizar la actualización generar una nueva lista (utilizando el TAD alumno), llamada Lista de Alumnos Desaprobados, la cual contiene los alumnos que su nota de desempeño no superó los 6 puntos, ordenada por nombre del alumno.

Nota: las estructuras utilizadas para resolver el problema deben estar definidas. El DNI de la Lista de Parciales seguro existe en la Lista Principal de Alumnos.

3.- Dado el siguiente TAD (**el cual no debe implementarse**)

```
TAD Insumo;
interface
    type exportado Insu;
    procedure CrearInsumo (Var i: insu; nom:string; precio:real; descripción:string; categoria: integer);
    procedure verNombre ( i: insu; var nom:string);
    procedure verDescripcion ( i: insu; var descripcion:string);
    function verPrecio (i: insu): real;
    function verCategoría (i: insu): integer;
    procedure modNombre (var i: insu; nom:string);
    procedure modDescripcion ( var i: insu; descripcion:string);
    procedure modPrecio (var i: insu; pre: real);
    procedure modCategoria (var i: insu; cat: integer);
    procedure asignar ( var i1: insu; i2:insu);
```

Una Facultad posee una estructura que almacena los insumos que compra a diferentes empresas. Para ello de cada empresa conoce el nombre, la dirección y todos los insumos que la empresa le provee ordenados por nombre.

Se pide:

- **a)** Generar una estructura donde para cada empresa aparezcan los insumos que provee ordenados por precio.
- **b)** Utilizando la estructura generada en a):
  - **i.** Informar el nombre de la empresa que provee más insumos.
  - **ii.** Modificar el nombre de los insumos con categoria =10. El nuevo nombre para todos los insumos es “Papel A4”.
  - **iii.** Para cada empresa informar el nombre y descripción de los insumos con precio mayor que 25 pesos. Esta búsqueda debe ser lo más eficiente posible.

4.- Dado el siguiente TAD (**el cual no debe implementarse**)

TAD **Tviaje**
interface
type exportardo viaje;

```
procedure crearUnViaje(var v: viaje, codigoAuto: integer; km: integer; dir1, dir2: string);
//Crea un viaje v realizado por el auto “códigoAuto” con distancia recorrida “km” con dirección origen “dir1” y dirección destino “dir2”
function verCodigoAuto(v: viaje): integer;
//Devuelve el código del auto que realizó el viaje “v”
function verKilometros(v: viaje):  integer;
//Devuelve la cantidad de kiometros que realizó el viaje “v”
procedure asignarViaje(var v1: viaje, v2: viaje);
//Asigna el viaje “v2” al viaje “v1”
```

Utilizando el TAD Tviaje realice el siguiente programa:
Una remisería dispone una lista de la información de los viajes (del tipo exportado del TAD) realizados durante el mes de septiembre de 2009. Esta información se encuentra ordenada por código de auto y para un mismo código de auto pueden existir 1 o más viajes.

- **a)** Realizar un módulo que reciba la lista con la información de los viajes realizados y genere una nueva estructura con la siguiente información: código de auto, total de kilometraje recorrido y cantidad de viajes realizados por el auto. Esta estructura debe estar ordenada por kilometraje recorrido y debe ser eficiente para la búsqueda por este criterio de ordenación. (Tener en cuenta, que para cada auto el total de kilómetros recorrido se calcula totalizando el kilometraje de cada viaje que realiza el auto).
- **b)** Luego de generada la estructura en el punto anterior y a partir de las estructuras anteriores utilice la más adecuada para calcular e informar:
  - **i.** Cantidad de autos con cantidad total de kilómetros recorridos entre 50.000 y 100.000.
  - **ii.** El auto con más viajes realizados.

NOTA: Modularizar. Declarar todas las estructuras de datos utilizadas. Realizar un programa que simule el llamado a los módulos realizados.

5.- Dado el siguiente TAD (**el cual no debe implementarse**)

TAD Tmesa;
interface
type exportado mesa;
procedure crearMesa(var m: mesa; capacidad: integer; numero: integer; precio: real);
//Crea una mesa "m" que se encuentra libre, con capacidad "capacidad", con número "numero" y precio "precio"
procedure asignarMesa(var m1: mesa; m2: mesa);
//Asigna mesa "m2" a la mesa "m1"
procedure ocuparMesa (var m: mesa);
//Marca la carpa "m" como ocupada
procedure liberarMesa (var m: mesa);
//Marca la carpa "m" como libre
function estaLibre (m: mesa): boolean;
//Retorna verdadero si la mesa "m" está libre
function verPrecio (m: mesa): real;
//Retorna el precio de la mesa "m"
prodecure actualizarPrecio (var m: mesa; p: real);
//Actualiza el precio de la mesa "m"
function verNumero (m: mesa): integer;
//Retorna el número de la mesa "m"
function verCapacidad (m: mesa): integer;
//Retorna la capacidad de la mesa "m"

Un restaurante dispone de una lista con la información de las mesas que posee para ser ocupadas (del tipo exportado de **Tmesa**). Esta información no posee orden alguno.
Se pide:

a) A partir de la información de las mesas que se dispone, generar una nueva estructura eficiente ordenada por capacidad de las mesas, donde para cada capacidad se tengan aquellas mesas de dicha capacidad.
b) Una vez genera la estructura anterior, realice un módulo que reciba dicha estructura y retorne el número de la mesa más costosa cuya capacidad se encuentra entre 5 y 10.
c) Realizar un módulo que reciba la estructura generada en a) y un parámetro que representa una capacidad y actualice el precio de todas las mesas de la capacidad recibida como parámetro, sumando $10 al precio actual de cada mesa que este libre.

6.- Implemente el TAD Pila de enteros .
Utilizando dicho TAD realice un programa para que lea 2 números de teclado e informe si son simétricos.

Ejemplo: 123456 y 654321 son simétricos

7.- Utilizando el TAD Pila de enteros, escriba un programa que lea números enteros hasta que se lee el número 0 e informe la cantidad de números capicúas.

8.- Utilizando el TAD conjunto**, implementado en la teoría**, crear programas para

a) Crear dos conjuntos y realizar la unión entre los mismos. Luego imprimir el resultado de la unión.

Ejemplo: A ={1,3,5,8} y B={3,4,5,9} AUB={1,3,4,5,8,9}

b) Crear dos conjuntos y suprimir de ambos los elementos que son el resultado de la intersección entre ellos.

Ejemplo: C={1,3,15,25} y D {1,3,4,5,6,7} resultado C={15,25} y D {4,5,6,7}

c) Crear dos conjuntos, compararlos y crear un tercer conjunto con los elementos que no son comunes a ambos.

Ejemplo: X={0,3,8,15,18,19,20} y Y {1,2,3,4,6,7,20,22,} resultado Z={0,1,2,4,6,7, 5,18,19,22}