HOJA 7.II. TADS Y PUNTEROS

1) Se desea realizar una aplicación para gestionar los vehículos de un concesionario de coches de segunda mano. Para ello, disponemos del `TAD Vehículo` para poder trabajar con información relacionada con los vehículos (la información que se maneja de cada vehículo es la matrícula, modelo y el precio de venta). El género de ese TAD se llama `Vehículo` y la especificación en seudocódigo contiene, entre otras, las siguientes operaciones:

```
acción iniciarVehículo (ent matrícula:cadena; ent
modelo:cadena; ent precio:real; sal v:Vehículo)
{Inicia v como un vehículo cuya matrícula es matrícula, cuyo
modelo es modelo y cuyo precio de venta es precio }
acción matrículaVehículo (ent v:Vehículo; sal
matrícula:cadena)
{matrícula toma como valor la matrícula del vehículo v }
acción modeloVehículo (ent v:Vehículo; sal modelo:cadena)
{modelo toma como valor el modelo del vehículo v }
función precioVenta (v:Vehículo) devuelve real
 {Devuelve el precio de venta del vehículo v}
acción modificarPrecio (e/s v:Vehículo; ent precio:real)
 {Modifica el precio de venta del vehículo v poniendo precio
 como nuevo precio de venta }
acción copiar (ent v1:Vehículo; sal v2:Vehículo)
{Inicia el vehículo v2 con los datos del vehículo v1}
```

a) **Especificar** un TAD CONCESIONARIO_SEGUNDA_MANO que permita modelar el concesionario y llevar a cabo su gestión. Las operaciones que nos interesa tener disponibles son:
   i. Crear el concesionario.
   ii. El dueño del concesionario compra un nuevo vehículo que incrementa el stock del concesionario.
   iii. El concesionario realiza la venta de un determinado vehículo.
   iv. Saber si el concesionario dispone o no de un vehículo de un determinado modelo.
   v. Saber el precio del vehículo más barato de un determinado modelo.
   vi. Obtener los datos de un vehículo determinado.

b) Dar una representación dinámica del TAD CONCESIONARIO_SEGUNDA_MANO.

c) **Implementar**, utilizando la representación del apartado anterior, las operaciones del TAD CONCESIONARIO_SEGUNDA_MANO y dar el coste computacional de cada una de ellas.

2) Disponemos del TAD TFG para poder trabajar con información relacionada con los trabajos fin de grado defendidos por los alumnos de una determinada titulación. El género de ese TAD se llama TFG y la especificación en seudocódigo contiene, entre otras, las siguientes operaciones:

```
acción iniciarTFG (ent titulo:cadena; ent autor:cadena; ent
director:cadena; ent calificacion:real; sal T:TFG)
{Inicia T como un TFG del alumno cuyo nombre es autor,
dirigido por el profesor cuyo nombre es director que lleva
por título titulo y que ha obtenido la nota calificacion. }
acción nombreAlumno (ent T:TFG; sal autor:cadena)
 {autor toma como valor el nombre del alumno que realizó el
 trabajo fin de grado T. }
acción nombreDirector (ent T:TFG; sal director:cadena)
 {director toma como valor el nombre del director del trabajo
 fin de grado T. }
función notaTFG (T:TFG) devuelve real
 {Devuelve la calificación obtenida por el trabajo fin de
 grado T.}
acción asignar (ent T1:TFG; sal T2:TFG)
{Inicia el trabajo fin de grado T2 con los datos del trabajo
fin de grado T1. }
```

a) **Especificar** un TAD CATALOGO_TFGs para gestionar los trabajos fin de grado defendidos en la titulación de GII. Las operaciones que nos interesa tener disponibles son:
   i. Crear un catálogo vacío.
   ii. Se ha defendido un nuevo TFG, añadirlo al catálogo de TFGs defendidos.
   iii. Saber si en la titulación se ha defendido o no algún TFG.
   iv. Saber qué alumno es el que mayor calificación ha obtenido. Si existen varios alumnos con la misma nota es suficiente con obtener uno de ellos.
   v. Saber el número de TFGs dirigidos por un determinado profesor.
   vi. Saber si un alumno ha defendido o no su TFG.

b) Dar una representación dinámica del TAD CATALOGO_TFGs.

c) **Implementar**, utilizando la representación del apartado anterior, las operaciones del TAD CATALOGO_TFGs y **dar el coste computacional** de cada una de ellas.

3) Disponemos del `TAD Calificación` para poder trabajar con información relacionada con las calificaciones de los alumnos de una determinada asignatura. El género de ese TAD se llama `CalAlumno` y la especificación en seudocódigo contiene, entre otras, las siguientes operaciones (su significado es el natural):

```
acción crearCalAlumno (ent nombre:cadena; ent NIF:cadena;
          ent calif:real; sal A:CalAlumno)
acción nombre (ent A:CalAlumno; sal nom:cadena)
acción NIF (ent A:CalAlumno; sal nif:cadena)
función nota (A:CalAlumno) devuelve real
acción modificarCalAlum (ent/sal A1:CalAlumno; ent calif:real)
acción copiaCalAlum (ent A1:CalAlumno; sal A2:CalAlumno)
acción mostrarCalAlum (ent A:CalAlumno)
```

a) **Especifica** un TAD CLASE (con género Clase) para gestionar las calificaciones obtenidas por los alumnos que cursan una asignatura. Las operaciones que nos interesa tener disponibles son:

i. Iniciar una clase a vacía (sin alumnos calificados).
ii. Añadir una calificación de un alumno a una clase.
iii. Modificar la calificación de un alumno ya almacenada (piensa en los parámetros que convienen)
iv. Saber si en una clase hay alguna calificación almacenada.
v. Número de alumnos calificados.
vi. Obtener un listado de las calificaciones de la clase, ordenado por el nombre de los alumnos.

b) Da una **representación dinámica** del TAD CLASE

c) **Implementa**, utilizando la representación del apartado anterior, las operaciones del TAD CLASE y **dar su coste computacional**.

4) Disponemos del TAD `Libro` para poder trabajar con información relacionada con libros. El género de ese TAD se llama `libro` y la especificación en seudocódigo contiene, entre otras, las siguientes operaciones (su significado es el natural):

```
acción crearLibro (ent titulo:cadena; ent ISBN:cadena;
     ent autor:cadena; ent precio: real; sal L:libro)
acción tituloL (ent L:libro; sal tit:cadena)
acción autorL (ent L:libro; sal aut:cadena)
acción ISBNL (ent L:libro; sal isbn:cadena)
función preiocL(L: libro) devuelve real
acción copiaL(ent L1:libro; sal L2:libro)
```

Usamos el TAD anterior y diseñamos el TAD Bibliografía (Lista de libros) con las siguientes operaciones:

```
acción IniciarBib (sal B: bibliografía)
acción AgnadirLibroBib (ent/sal B: bibliografía; ent L:libro)
acción EliminarLibroBib(ent/sal B:bibliografía; ent
isbn:cadena)
función NumLibros(B:bibliografia) devuelve entero
función PosicionLibro(B:bibliografía; isbn:cadena) devuelve
entero
función libroEnPos(B:bibliografía; pos:entero) devuelve libro
acción MostrarBib (ent B: bibliografía)
```

a) Da una posible "representación dinámica" para una Bibliografía.
b) Implementa, usando la representación que hayas propuesto, la función PosiciónLibro (si el libro buscado no está devuelve cero).
c) **Usando el TAD** Bibliografía, diseña una acción para eliminar de una bibliografía el libro más caro.