Práctico 4: Tipos de Listas y Lista de listas

1. Defina un tipo de datos para una Fila, implementándola como una lista vinculada. Los elementos son números enteros. Luego codifique las funciones / procedimientos implementados en el "ESTRUCTU" para la fila:

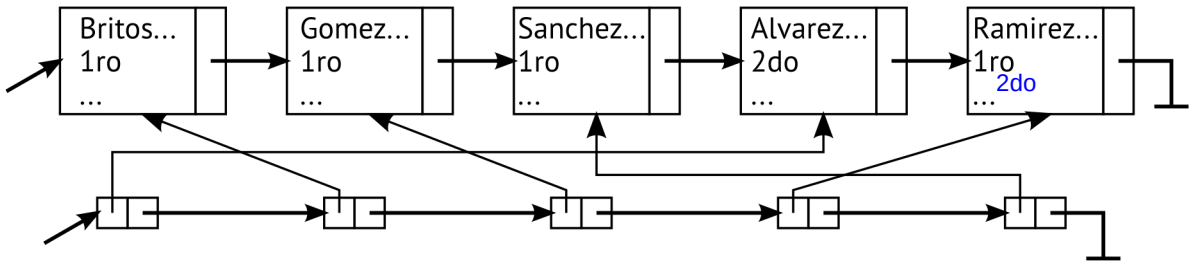
`Inic_fila`(no es necesario implementarlo)	`Fila_vacia`	`Write_fila`
`Agregar`	`Primero`	`Read_fila` (no es necesario implementarlo)
`Extraer`		

2. Codifique un módulo que dada una lista ordenada doblemente vinculada de reales y un número que representa una posición, elimine el elemento correspondiente a esa posición

3. Codifique un módulo que elimine, si existe, un número de una lista circular de enteros.

4. Se tiene una lista vinculada de alumnos en el que cada nodo posee los siguientes datos:
   - Apellido y Nombre
   - Curso (1ro, 2do, 3ro, 4to , 5to ó 6to)
   - DNI
   - Domicilio
   - Teléfono

   La lista se encuentra ordenada por curso y dentro de cada curso por apellido. Para obtener un listado alfabético de todos los alumnos implementaremos una lista "invertida". Los nodos de una lista invertida apuntan a los nodos de la lista original pero se vinculan entre sí en otro orden, como en el siguiente gráfico:



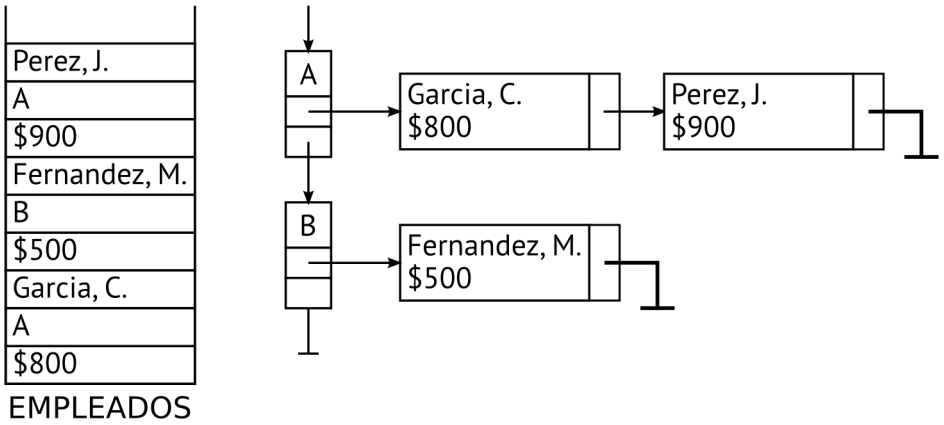
   Si recorremos la lista invertida, el orden de los nodos es alfabético (Alvarez, Britos, Gomez, Ramirez, Sanchez) en lugar de por curso y esto nos permite emitir el listado con una pasada.

   Codifique el procedimiento que, dada la lista vinculada original, construya y devuelva la lista invertida correspondiente.

   b) Codifique un procedimiento que dé de alta un nuevo alumno en la estructura anterior, actualizando las dos listas para que ambas mantengan el orden que corresponda. Para simplificar suponga que el procedimiento recibe como parámetro el nodo con los datos del nuevo alumno.

5. Se tiene un archivo EMPLEADOS cuyos elementos son registros con los datos de un operario de una fábrica. Los datos son los siguientes:
   - Apellido y nombre
   - Categoría
   - Sueldo

   a) Codifique un programa que lea del archivo EMPLEADOS para generar una lista ordenada de categorías, donde, para cada categoría, existe otra lista con los nombres y sueldos de los empleados de la misma ordenada por apellido.



b) Codifique un procedimiento que, dada la lista anterior, pida un número de categoría e imprima todos los empleados que la poseen.
c)
d) Codifique un procedimiento que, dada una categoría, si existe de la elimine de la lista anterior, junto con todos sus empleados.

6) Retome el ejercicio de la lista de Jugadores(el Nro 7 del práctico 3). Modifique el registro para que contenga otro puntero llamado Sig_puntaje (del mismo tipo) que permitirá recorrer la lista ordenada de manera ascendente por puntaje:

```
Type PuntJugador = ^ TipoJugador;

TipoJugador = Record

              Alias: String

              Puntaje:0..100000;

              Sig:PuntJugador ;

              Sig_Puntaje:PuntJugador;

              End;
```

Realizar un módulo que reciba como parámetro la Lista Jugadores y que deberá actualizar los punteros Sig_puntaje (inicialmente en nil) para que se pueda recorrer la lista ascendentemente por puntaje. El nuevo orden se recorrerá a partir del puntero Jugadores_Puntaje, inicialmente en nil y recibido también como parámetro.