Nota: los ejercicios no están corregidos ni compilados.

1. Cargar un vector **de hasta** 100 números enteros. Implementar un módulo que elimine todas las ocurrencias de un determinado número que se recibe como parámetro en ese módulo.

```
program ej1;
type
    vector = array[1..100] of integer;

procedure eliminar(numero: integer; var v:vector; var dl : integer)
var
    i : integer;
begin
    i := 1;
    while (i < dl) do
        if (v[i] = numero) then
        begin
            for j := i to dl-1 do
                v[j] := v[j+1];
            dl := dl - 1;
        end;
        i := i + 1;
end;
```

3. Detallar (realizando la descripción de cada paso) cómo se realizaría la inserción de un valor entero dentro de un vector ordenado de 10 elementos enteros, considerando que si el vector ya está totalmente ocupado quedarán los 10 elementos de mayor valor.

```
program ej2;
const
    df = 100

procedure InsertarOrdenado(var v:vector;numero: integer; var dl:integer)
var
    i,j :integer;
begin
    i := 1;
    while (v[i] < numero) and (i < dl) do i := i+1; //Recorro el vector buscando la posición
    if (v[i] < numero) then //Si debo insertarlo
        if (dl < df) then //Si tengo espacio debo agregarlo
            begin
                dl := dl +1;
                for j := dl downto i do v[j] := v[j-1]; //muevo todos los elementos una posic
            end
        else //si no hay espacio y el numero es mayor entonces lo reemplazo por el ultimo
            v[dl] := numero;
end;
```

4. Se desea procesar información de personas anotadas en una maestría. De cada una se conoce el apellido, nombre, dirección y especialidad. Se pide generar una lista ordenada por apellido.

```
program ej4
{
type
    persona = record
        apellido : string;
        nombre : string;
        direccion : string;
        especialidad : string;
    end;
    lista = ^.nodo
    nodo = record
        d : persona;
        s : lista;
    end;
procedure insrtarOrdenado(pri : lista; p : persona)
var
    ant,act,nue : lista;
begin
    new(nue);
    nue^.d := p;
    act := pri;
    ant := pri;
    while (act <> nil) and (p.apellido < act^.d.apellido) do
    begin
        ant := act;
        act := act^.s;
    end;
    if (ant = act) then
        pri := nue
    else
        ant^.s := nue;
    nue^.s := act;
end;
procedure leerPersona(var p : persona)
begin
    read(p.apellido)
    if (p.apellido <> 'fin') then
    begin
        read(p.nombre);
        read(p.direccion);
        read(p.especialidad);
    end;
end;

var
    l,nue,aux :lista;p : persona;

begin
    l := nil;
    leerPersona(p);
    while (p.apellido <> 'fin')do
    begin
        insertarOrdenado(l,p);
        leerPersona(p);
    end;
end.
```

5. Un comercio dispone de las ventas realizadas para sus productos. De cada venta se conoce número de producto (1..300), cantidad vendida y nombre de producto. Además el comercio cuenta con una tabla con el precio por unidad de cada uno de los 300 productos. Se pide calcular e informar el nombre del producto con el cual el comercio obtuvo la menor ganancia.
Notas: las ventas están ordenadas por número de producto. Un producto puede ser vendido 0, 1 o más veces.

```
program ejRecorrerLista
{
type
    venta = record
        numero : (1..300);
        cantidad : integer;
        nombre : string;
    end;
    lista = ^.nodo;
    nodo = record
        d : venta;
        s : lista;
    end;
    table = Array[1..300] of real;
var
    nombreMin,nombreAct : string;
    gananciaMin,gananciaAct : real;
    l : lista;
    t : tabla;
begin
    while (l <> nil) do
    begin
        nombreAct := l^.d.nombre;
        gananciaAct := 0;
        while (nombreAct = l^.d.nombre) and (l <> nil) do
            gananciaAct := gananciaAct + l^.d.cantidad * t[l^.d.numero];
            l := l^.s;
        if (gananciaAct < gananciaMin) then
        begin
            gananciaMin := gananciaAct;
            nombreMin := nombreAct;
        end;
    end;
    write(nombreMin);
end.
```