```
NOTA GENERAL DEL FINAL LIBRE (10)
OTRAS ENTREGAS
ESTAN BIEN, LAS FUNCIONES EXISTE QUE RETORNAN BOOLEANOS LAS REEMPLAZARÍA POR
FUNCIONES QUE RETORNAN UN PUNTERO (CUMPLE DOBLE FUNCION, VER SI EXISTE Y MODIFICAR SOBRE EL
MISMO)

Program Parcial;
//BIEN
Type PuntArbol = ^NodoArbol;
    NodoArbol = Record
        Nro_Socio,DNI:integer; // orden por nro_socio
        Menor,Mayor:PuntArbol;
    end;
    PuntLista = ^NodoLista;
    NodoLista = Record // orden descendente por DNI
        Puntero:PuntArbol;
        Sig:PuntLista;
    end;
//BIEN
procedure CargarNodoLista(var nuevo:puntlista; nodo:puntarbol);
begin
    new(Nuevo);
    Nuevo^.Sig:= Nil;
    Nuevo^.Puntero:= Nodo;
end;
//BIEN
Procedure VincularNodoOrdenado(var Lista:PuntLista; Nuevo:PuntLista);
// Dado un nodo lo inserta descendientemente en la lista
var CursorAux:PuntLista;
begin
    If (Lista = Nil) then begin // Primero caso, que no hayan datos en la lista y sea el
    primero elemento.
        Lista:= Nuevo;
        Lista:= Lista^.Sig;
    end
    else
        CursorAux:= Lista;
        If Lista^.Puntero^.DNI > Nuevo^.DNI then begin // Segundo caso, que el dni ingresado
        corresponda hacia la derecha
            While (CursorAux^.Sig <> Lista)  do
                CursorAux:= CursorAux^.Sig;
            Nuevo^.Sig:= CursorAux;
            Lista:= Nuevo;
        end
        else // tercer caso, que el dni ingresado sea el mayor de todos
            While (CursorAux^.Sig <> Lista) and (CursorAux^.Puntero^.DNI <= Nuevo^.DNI) do
                CursorAux:= CursorAux^.Sig;
            Nuevo^.Sig:= CursorAux^.Sig;
            CursorAux^.Sig:=Nuevo;
end;
//BIEN
Procedure BuscaNodos(Arbol:PuntArbol;var Lista:PuntLista;Nivel_Actual,MinNivel,MaxNivel,MaxSocio
,MinDNI:Integer);
// Modulo encargado de comparar todos los requisitos para ser creada la lista
var nuevo:PuntLista;
begin
    If Arbol <> Nil then begin
        If Arbol^.Nro_Socio < MaxSocio then begin // Primer cota, ya que se ordena por
        Nro_Socio el arbol asi no recorro de mas el arbol innecesariamente
            BuscaNodos(Arbol^.Menor,Lista,(Nivel_Actual+1),MinNivel,MaxNivel,MaxSocio,MinDNI);
            BuscaNodos(Arbol^.Mayor,Lista,(Nivel_Actual+1),MinNivel,MaxNivel,MaxSocio,MinDNI);
            If (Arbol^.DNI > MinDNI) and ((Nivel_Actual < MinNivel) and (Nivel_Actual > MaxNivel
            )) then begin
                CargarNodoLista(Nuevo,Arbol);
                VincularNodoOrdenado(Lista,Nuevo);
        end
```

```
        else
            If Arbol^.Nro_Socio > MaxSocio then //BIEN
                BuscaNodos(Arbol^.Menor,Lista,(Nivel_Actual+1),MinNivel,MaxNivel,MaxSocio,MinDNI
                );
    end;
end;
var ArbolSoc:PuntArbol;
    ListCirc:PuntLista;
    MinNivel,MaxNivel,MaxSocio,MinDNI:integer;
begin
    Writeln('Ingrese MinNivel');
    readln(MinNivel);
    Writeln('Ingrese MaxNivel');
    readln(MaxNivel);
    Writeln('Ingrese MaxSocio');
    readln(MaxSocio);
    Writeln('Ingrese MinDNI');
    readln(MinDNI);
    BuscaNodos(ArbolSoc,ListCirc,0,MinNivel,MaxNivel,MaxSocio,MinDNI);
end.
```