Ejercicio 1 (35 puntos)

Las siguientes definiciones de constantes y tipos representan las palabras que componen un texto y sus valores de polaridad. Estos valores son neutro (por ejemplo, "mesa"), positivo (por ejemplo, "feliz") o negativo (por ejemplo, rechazo").

```
const
   Max = ...; (* valor mayor que 1 *)

type
   Polaridad = record
                  case esNeu:Boolean of
                     false: (valor: (pos, neg));
                     true: ()
               end;
   Palabra = record
                pal: Cadena;  (* se asume definido, no se usa *)
                pol: Polaridad
             end;
   Texto = record
              pals: array [1..Max] of Palabra;
              tope: 0..Max
           end;
```

a) (15 puntos)

Implemente el procedimiento:

```
procedure polaridadTexto(tex : Texto; var polTex : Polaridad);
```

Este procedimiento calcula la polaridad del texto y la devuelve en el parámetro polTex. La polaridad del texto se establece según la polaridad predominante entre las palabras que lo componen:

- si la cantidad de palabras positivas es mayor que la de negativas y también mayor que la de neutras, el texto es positivo
- si la cantidad de palabras negativas es mayor que la de positivas y también mayor que la de neutras, el texto es negativo.
- en cualquier otro caso el texto es neutro. Asuma que el texto tiene al menos una palabra.

b) (20 puntos)

Implemente la función:

```
function alMenosUno(tex : Texto) : Boolean;
```

Esta función devuelve TRUE si en el texto hay al menos una palabra con cada polaridad y FALSE en caso contrario.

Ejercicio 2 (20 puntos)

Dada la siguiente declaración:

```
const   M = ...; (* valor mayor que 1 *)
type    arr = array [1..M] of integer;
```

Implemente la función:

```
function esSumaSiguientes (var a: arr): integer;
```

Esta función recibe como parámetro un arreglo y retorna el índice del arreglo donde se encuentra un elemento que sea igual a la suma de todos los siguientes. En caso de que no exista, devuelve 0. Por ejemplo, en el arreglo [1, 3, 4, 13, 4, 6, 3] La función deberá retornar 4, ya que en esa posición se encuentra el número 13, que es la suma de los siguientes (4+6+3=13). Puede asumir que a lo sumo existe un elemento que cumpla esta condición.

Ejercicio 3 (25 puntos)

Dada la siguiente declaración:

```
type
    Lista = ^TipoCelda;
    TipoCelda = record
                    dato: integer;
                    sig: Lista
                end;
```

Implemente el procedimiento:

```
procedure intercalarListas(var l1 : Lista; var l2 : Lista; var l3 : Lista);
```

Este procedimiento recibe dos listas del mismo largo (`l1` y `l2`), y retorna en `l3` los elementos de `l1` y `l2` intercalados. No debe crear memoria nueva sino utilizar las celdas ya existentes en `l1` y `l2`. Al terminar el procedimiento, `l1` y `l2` deben ser `nil`.

Ejemplos:

Entrada:
`l1` = $\boxed{5} \to \boxed{3} \to \boxed{4} \to$ `nil` `l2` = $\boxed{6} \to \boxed{1} \to \boxed{8} \to$ `nil`
Salida:
`l1` = `nil` `l2` = `nil` `l3` = $\boxed{5} \to \boxed{6} \to \boxed{3} \to \boxed{1} \to \boxed{4} \to \boxed{8} \to$ `nil`

Entrada:
`l1` = $\boxed{5} \to$ `nil` `l2` = $\boxed{6} \to$ `nil`
Salida:
`l1` = `nil` `l2` = `nil` `l3` = $\boxed{5} \to \boxed{6} \to$ `nil`

Entrada:
`l1` = `nil` `l2` = `nil`
Salida:
`l1` = `nil` `l2` = `nil` `l3` = `nil`

Ejercicio 4 (10 puntos)

Diga cuál es la salida del siguiente programa, si se lee de la entrada el dígito posterior al guión de su cédula.

```
program alcance(input,output);

var num : integer;

function f (a: integer) : boolean;
var b : integer;
begin
   b := (3 * a) div 2;
   f := (a = num) or (b = num)
end;

procedure p (var c: integer);
begin
   c := (c + 3) div 5;
   writeln(c, num)
end;

begin
   readln(num);
   num := num + 2;
   p(num);
   writeln(num);
   if f(num) then
      num := num * 2
   else
      num := num * 3;
   writeln(num);
end.
```