EJERCICIO 1

Dado el siguiente programa, escribir cuál será su salida cuando la variable x se carga de la entrada estándar con el último dígito de su CI (antes del dígito verificador). Por ejemplo, si su CI es 1.234.567-8, el último dígito es 7:

```
Program Febrero;

var  x, y, z: integer;

Procedure uno (a : integer; var b : integer);
   Function tres (a, b : integer) : integer;
      var i : integer;
      begin
        for i := 1 to 3 do
            a := a + b;
        tres := a
      end;
   begin
      b := tres ( b, b);
      a := x * 3;
      writeln (a, b);
      z := a + 3
   end;
```

```
Function dos (b : boolean): integer;
  begin
     if b then
        dos := 5
     else
        dos := 3
  end;

begin
   readln(x);
   y := dos ( x > 5 );
   z := dos ( y > 3 );
   uno (x, y);
   writeln(x, y, z)
end.
```

Solución

Entrada	Salidas
1	3 12 1 12 6
2	6 12 2 12 9
3	9 12 3 12 12
4	12 12 4 12 15
5	15 12 5 12 18

Entrada	Salidas
6	18 20 6 20 21
7	21 20 7 20 24
8	24 20 8 20 27
9	27 20 9 20 30
0	0 12 0 12 3

EJERCICIO 2

Dadas las siguientes declaraciones:

```
const MAX = ...; (* entero mayor que 0 *)
type Numeros = array [1 .. MAX] of integer;
```

Escribir la siguiente función:

```
function repetidos(nums : Numeros) : boolean;
```

que devuelve TRUE si el arreglo *nums* contiene valores repetidos y FALSE en caso contrario.

Solución

```
function repetidos(nums: Numeros) : Boolean;
(* retorna TRUE  si el arreglo nums tiene valores repetidos *)

var  i, j : Integer;
     repetido : Boolean;

begin
   i := 1;
   repetido := FALSE;
   repeat
      j := i + 1;
      while (j <= MAX) and (nums[i] <> nums[j]) do
         j := j + 1;
      repetido := j <= MAX;
      i := i + 1
   until (i = MAX) or repetido;
   repetidos := repetido
end;
```

EJERCICIO 3

Sean las siguientes declaraciones:

```
CONST
   MAX_EST = ...; {valor mayor que 0}
   CANT_RESP = ...; {valor mayor que 0}
TYPE
    TipoPrueba = (escrita,oral);
    Respuestas = array[1..CANT_RESP] of Char;
    Prueba = record
                    ci: Integer;
                    case tipo : TipoPrueba of
                        escrita : (resp: Respuestas);
                        oral : (aprobo: Boolean);
             end;
     Pruebas = record
                        pruebas: array[1 .. MAX_EST] of Prueba;
                        tope: 0..MAX_EST;
                end;
```

En el arreglo de tipo Pruebas se almacenan las pruebas realizadas por alumnos de un curso, que pueden ser orales o escritas. Si son escritas se guardan las respuestas del estudiante en el arreglo ***resp***. Se asume que las respuestas posibles son 'a', 'b', 'c', 'd' o el carácter 'v' que indica respuesta vacía.
Implementar la función:

```
function puntaje(pc: Pruebas; ci: Integer; correctas: Respuestas):Real;
```

que retorna cuántos puntos sacó el alumno de cédula ci en la prueba escrita.

- En el parámetro **correctas** se tienen las respuestas correctas para cada pregunta.
- Cada respuesta correcta del alumno suma 2 puntos, las incorrectas restan 0.25, las vacías no afectan el puntaje.
- Si no existe un alumno con la cédula dada, o existe pero realizó una prueba oral, la función retorna el valor especial -1000.

Cada alumno aparece a lo sumo una vez en el arreglo pruebas.

Solución

```
function puntaje(prs: Pruebas; ci: Integer; correctas: Respuestas):
Real;
    var i,j : Integer; result: Real;

    begin

      {busco el alumno por su ci}
      i := 1;
      while (i <= prs.tope) and (prs.pruebas[i].ci <> ci) do
        i := i + 1;

      if (i <= prs.tope) and (prs.pruebas[i].tipo = escrita) then
      {si encontre el alumno y su prueba es escrita, calculo puntos}
      begin
         result := 0;
         for j := 1 to CANT_RESP do
            {si son iguales sumo 2}
            if (prs.pruebas[i].resp[j] = correctas[j]) then
               result := result + 2
            {si son distintas y la respuesta no es vacia resto 0.25}
            else if(prs.pruebas[i].resp[j] <> 'v') then
               result := result - 0.25
      end
      else
      {si no existe el alumno o su prueba es oral, devuelvo -1000}
         result := -1000;

      puntaje := result

    end;
```

EJERCICIO 4

Dadas las siguientes declaraciones:

```
TYPE
   ListaInt = ^Celda;
   Celda = RECORD
             info: Integer;
             sig: ListaInt
           END;
```

Escribir el procedimiento:

```
procedure p (var l : ListaInt; n : Integer);
```

que elimina las repeticiones consecutivas del entero n, dejando una única ocurrencia de n en l. Asuma que todas las ocurrencias de un número n aparecen en forma consecutiva en la lista l.

Solución

```
procedure p (var l : ListaInt; n : Integer);

var ls, aux : ListaInt;

begin
   ls := l;
   while (ls <> nil) and (ls^.info <> n) do
      ls := ls^.sig;
   if (ls <> nil) then
   begin
      while (ls^.sig <> nil) and (ls^.sig^.info = n) do
      begin
         aux := ls^.sig;
         ls^.sig := aux^.sig;
         dispose(aux)
      end
   end
end;
```