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 (el dígito verificador). Por ejemplo, si su CI es 1.234.567-8, el último dígito es 8:

```
program marzo;

var  x, y, z: integer;

   procedure proc (a : integer; var b : integer);
         function fun (a, b : integer) : integer;
            begin
               a := a * 2;
               fun := a + b
            end;
  var z : integer;
  begin (* proc *)
    z := a + 1;
    b := fun(a, b);
    y := z + b;
    writeln (a, b, z);
  end; (* proc *)

begin  (* principal *)
  readln(x);
  y := x + 2;
  z := y + 1;
  proc (x, y);
  writeln(x, y, z)
end.
```

Solución

Entrada	Salidas
1	1 7 2 1 7 4
2	2 11 3 2 11 5
3	3 15 4 3 15 6
4	4 19 5 4 19 7
5	5 23 6 5 23 8

Entrada	Salidas
6	6 27 7 6 27 9
7	7 31 8 7 31 10
8	8 35 9 8 35 11
9	9 39 10 9 39 12
0	0 3 1 0 3 3

EJERCICIO 2

Considere los tipos:

```
type
   TIndice = 1..MAX;
   TA      = array [TIndice] of integer;
   TOrden  = (creciente, decreciente, desordenado);
```

Considere que MAX > 1 y que todos los elementos del arreglo son diferentes.

Escriba la función:

```
function ordenado (A : TA): TOrden;
```

que determina si los elementos del arreglo A están ordenados en forma creciente, están ordenados en forma decreciente, o están desordenados.

Solución

```
function ordenado (A : TA): TOrden;
var i : integer;
   crece, decrece : boolean;
begin
   i := 1;
   crece   := false;
   decrece := false;
   while (i <MAX) and not (crece and decrece) do
   begin
      if A[i] < A[i+1] then
         crece:= true
      else
         decrece:= true;
      I := i + 1
   end;

   if crece and decrece then
      ordenado := desordenado
   else if crece then
      ordenado := creciente
   else
      ordenado := decreciente
end;
```

Otra posible solución

```
function ordenado (A : TA): TOrden;
var i :  integer;
begin
   if A[1] < A[2] then
   begin
      i := 2;
      while (i < MAX) and (A[i] < A[i+1]) do i := i + 1;
      if i = MAX then ordenado := creciente else ordenado := desordenado
   end
   else begin
      i := 2;
      while (i < MAX) and (A[i] > A[i+1]) do i := i + 1;
      if i = MAX then ordenado := decreciente else ordenado := desordenado
   end
end;
```

EJERCICIO 3

Se tienen los tipos:

```
type
   PosibleEntero = record
                     case vacio : boolean of
                       true : ();
                       false: (valor : integer);
                     end;

   TArreglo = array [1..MAX] of PosibleEntero;

   TArregloConTope = record
                       elementos : array [1..MAX] of integer;
                       tope      : 0 .. MAX
                     end;
```

Escriba el procedimiento:

```
procedure Defrag(arreglo : TArreglo; var resultado : TArregloConTope);
```

Este procedimiento copia todos los enteros que están en `arreglo` (en las celdas cuyo campo `vacio` es false) a la estructura `resultado`. Las celdas de `arreglo` cuyo campo `vacio` es true se ignoran. Los enteros en `resultado` deben aparecer en el mismo orden que aparecían en `arreglo`. Ejemplos (con MAX=10):

- arreglo = [(),3,5,(),8,2,3,(),7,4] resultado = (elementos=[3,5,8,2,3,7,4,...], tope = 7)
- arreglo = [(),(),(),(),(),(),(),(),(),()] resultado = (elementos=[...], tope = 0)
- arreglo = [10,3,5,2,8,2,3,19,7,4] resultado = (elementos=[10,3,5,2,8,2,3,19,7,4], tope = 10)

Solución

```
procedure Defrag( arreglo : TArreglo; var resultado :
TArregloConTope);
var
   i,j : integer;
begin
   j:= 0;
   for i:= 1 to MAX do
      if not arreglo[i].vacio then
      begin
         j:= j + 1;
         resultado.elementos[j]:= arreglo[i].valor
      end;
   resultado.tope:= j
end;
```

EJERCICIO 4

Dadas las siguientes declaraciones:

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

Escribir la función:

```
function deHastaSin (ini, fin : integer; sin : ListaInt) : ListaInt;
```

que retorna la lista que contiene los enteros en el rango entre ini y fin que no pertenecen a la lista sin. La lista resultante debe quedar ordenada de menor a mayor.

Ejemplos:

- si ini = 1, fin = 4 y sin = (), retorna (1,2,3,4).
- si ini = 1, fin = 4 y sin = (6,5), retorna (1,2,3,4).
- si ini = 1, fin = 4 y sin = (6,2,5), retorna (1,3,4).
- si ini = 1, fin = 4 y sin = (6,2,5,3,4,1), retorna ().
- si ini = 1, fin = 1 y sin = (6,5), retorna (1).
- si ini = 4, fin = 1 y sin = (6,5), retorna ().

Solución

```
function deHastaSin (ini, fin : integer; sin : ListaInt) : ListaInt;

   (* funcion auxiliar, busca a v en la lista l *)
   function enLista (v : integer; l : ListaInt) : boolean;
   var laux : ListaInt;
   begin
      laux := sin;
      while (laux <> nil) and (laux^.info <> v) do
         laux := laux^.sig;
      enLista := laux <> nil
   end;

var lista, laux : ListaInt;
    i   : integer;
begin
   lista := nil;

   for i := fin downto ini do
      if not (enLista(i,sin)) then
      begin
         new(laux);
         laux^.sig  := lista;
         laux^.info := i;
         lista := laux
      end;

   deHastaSin := lista
end;
```