*Duración total del examen:* ***3 horas***

# Ejercicio 1

[3.5 puntos]

Se desea hacer una distribución en grupos de prácticas de los estudiantes de una asignatura. Se dispone de un listado en un fichero de registros, ya ordenado por apellidos-nombre y cuyo formato es el que se indica en las notas.

Se quiere hacer una agrupación por iniciales (por ejemplo, en el grupo 1 están los estudiantes cuyos apellidos comienzan por A, B y C; en el grupo 2, los que empiezan por D y E;….). Para hacer una distribución más o menos homogénea, tenemos que saber cuántos apellidos empiezan con cada letra del alfabeto. Suponemos que no hay más de 15 estudiantes con la misma inicial.

Ahora hay que proceder a la agrupación. El criterio es el siguiente: se empieza a llenar el grupo 1 con los estudiantes de la A, la B,… hasta que sumando los de la siguiente inicial se rebasa el número de 15; ésta será la primera inicial del segundo grupo. Así sucesivamente hasta agotar la lista.

Se pide:

1. Definir las estructuras de datos que permitan gestionar los datos.
2. Desarrollar el procedimiento
   ```
   Procedure CuantosDeCadaInicial(??? fAl: tpFichero; ??? vIn: tpIniciales)
   {lee los datos del fichero "fAl" y devuelve en "vIn" una lista con el número de apellidos que empiezan por cada inicial}
   ```
3. Desarrollar el procedimiento
   ```
   Procedure GruposDePracticas(??? fAl: tpFichero; ??? fGrupos: text);
   {lee los datos del fichero "fAl" y devuelve en el fichero fGrupos la lista de los grupos de prácticas con el formato mostrado en el ejemplo}
   ```
4. Desarrollar el programa principal que, utilizando todo lo anterior, solicite por teclado (entrada estándar) el nombre de los dos ficheros y genere el fichero de texto.

**Notas:** Los ficheros no contienen errores. No hay ningún apellido con inicial Ñ ni acentuada.Todos los apellidos empiezan por mayúscula.

**Estructuras de datos (deben respetarse)**

```
Const L_CAD = 50; L_DNI = 9;
Type
   tpDNI = Array[1..L_DNI] Of char;
   tpVectCar = Array[1..L_CAD] Of char;
   tpCadena = Record
      long: 0..L_CAD;
      cars: tpVectCar
   End;
   tpDato = Record
      NIP: integer;
      DNI: tpDNI;
      apel1,apel2,nom: tpCadena
   End;
   tpFichero = File Of tpDato;
```

**Ejemplo: contenido del fichero de grupos**

```
Grupo 1: A B C 13 alumnos
Grupo 2: D E F G H 14 alumnos
Grupo 3: I J K 8 alumnos
Grupo 4: L 11 alumnos
Grupo 5: M 14 alumnos
Grupo 6: N O P Q 13 alumnos
Grupo 7: R S T 12 alumnos
Grupo 8: U V W X Y Z 6 alumnos
```

# Ejercicio 2

[3.0 puntos]

El Documento Nacional de Identidad, DNI, consta de un número de 8 cifras seguido de una letra. La letra se obtiene a partir de las cifras como sigue: se calcula el resto de dividir el número por 23, que dará un número entre 0 y 22. La letra correspondiente es la asociada al resto de la división de acuerdo a la siguiente tabla:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | R | W | A | G | M | Y | F | P | D | X | B | N | J | Z | S | Q | V | H | L | C | K | E |

Los datos de esta tabla están guardados en un fichero de texto denominado `'tabla.txt'`, de manera que en cada línea hay un par formado por una letra y su valor asociado, separados por blancos.

El ejercicio pide:
1) Desarrollar el código de la función DNICorrecto:

| **Especificación de la función (debe respetarse)** | **Ejemplo: contenido del fichero** 'tabla.txt' |
|---|---|
| **Const** L_DNI = 9;<br>**Type** tpDNI = **Array**[1..L_DNI] **Of** char;<br>**Function** DNICorrecto(??? DNI: tpDNI): **boolean**;<br>{Pre: "DNI" contiene 8 dígitos, y una de las 22 letras en su posición 9<br>Post: determina si "DNI" es correcto de acuerdo a las reglas establecidas<br>} | Y 6<br>H 18<br>. . .<br>T 0<br>E 22<br>P 8<br>F 7<br>Z 14 |

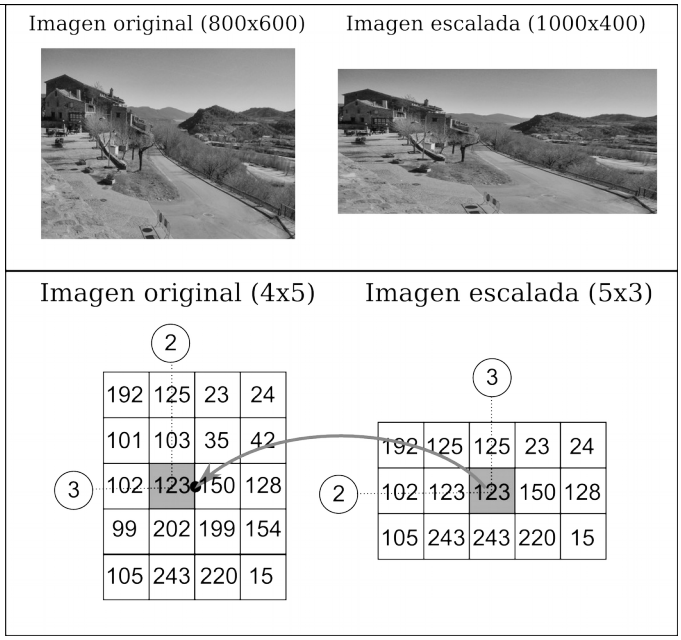
2) Escribir un programa Pascal que lea de la entrada estándar (teclado) un DNI e informe por la salida estándar (pantalla) si es correcto o no.

**Nota:** Podemos asumir que el compilador de Pascal usado contiene el tipo "longint", cuyo dominio de valores es el rango (-2147483648..2147483647), aunque se valorará trabajar con datos de tipo "integer", cuyo dominio es (-32768..32767).

## Ejercicio 3 [3.5 puntos] [3.5 puntos]

Una imagen digital no es más que una matriz bidimensional de elementos llamados píxeles, cuyos valores numéricos indican el color de la correspondiente región de la foto. En el caso de una imagen en escala de grises, estos valores son números naturales que van desde 0 (que indica negro) hasta 255 (que indica blanco). Las imágenes (en este ejercicio) tendrán un tamaño máximo de 1000x1000.

Se desea poder *escalar la imagen* (cambiar su tamaño) tanto a imágenes más grandes como más pequeñas, sin preservar la relación entre anchura y altura de la imagen. Para ello la imagen escalada (nueva) debe rellenar cada píxel con un valor obtenido de la imagen original en la posición correspondiente. Cada pixel de la imagen nueva toma el valor correspondiente al pixel de la original cuya posición se obtiene aplicando los correspondientes factores de escala en horizontal y vertical.




Se pide:

1. Desarrollar el tipo de datos `tpImagen` que permita guardar una imagen en memoria.
2. Desarrollar el siguiente procedimiento que, dada una posición sobre la imagen escalada, obtenga una posición sobre la imagen original.

```
Procedure Posicion(??? cEsc, fEsc:Integer; ??? cOri, fOri:Integer;
                   ??? anchEsc, altEsc:Integer; ??? anchOri, altOri:Integer);
{Se devuelve en "cOri" y "fOri" la columna y la fila (respectivamente) de la imagen
original, en la que se buscará el contenido del píxel de columna "cEsc" y fila "fEsc"
de la imagen escalada. Las dimensiones de la imagen original son "anchOri" x "altOri" y
las de la imagen escalada son "anchEsc" x "altEsc". }
```

3. Desarrollar el siguiente procedimiento que escala una imagen, utilizando el procedimiento anterior.

```
Procedure Escala(??? ori:tpImagen; ??? esc:tpImagen; ??? anch, alt: integer);
{"ori" contiene la imagen original. "anch" y "alt" tienen valores entre 1 y 1000. "esc"
contendrá la imagen original escalada, con la anchura y la altura que se pasan como
parámetros. }
```