Examen de **Programación 14 de Enero de 2008.**
Dpto. Informática e Ingeniería de Sistemas
Centro Politécnico Superior. Universidad de Zaragoza

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

Ejercicio 1 [3 puntos]

El fichero de texto '*numeros.txt*' contiene exclusivamente números naturales separados por uno o más espacios en blanco (tras el último número no hay ningún espacio en blanco). Por supuesto, este fichero puede tener una o más líneas y, a priori, se desconoce el número de enteros que están almacenados en él, pudiendo ser una cantidad muy grande.

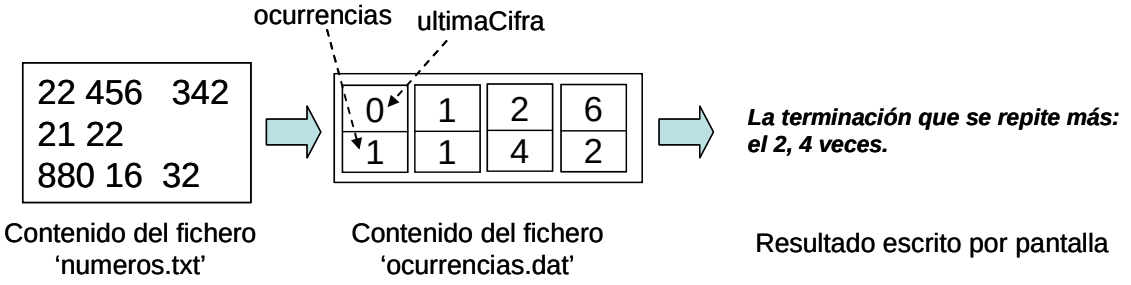
Se pide:

a) Desarrollar un programa PASCAL que lea el fichero anterior y calcule cuántos números tienen la misma última cifra, es decir, cuántos números finalizan en cero, uno, dos, y así sucesivamente, hasta nueve. El resultado de este cálculo deberá almacenarse en un fichero de registros de tipo `tpOcurrenciaNum` llamado *'ocurrencias.dat'*. Cada registro concreto almacena el número de ocurrencias (campo `ocurrencias`) encontradas para una última cifra concreta (campo `ultimaCifra`). Sólo se almacenará en el fichero información sobre aquellas últimas cifras que tengan una o más ocurrencias.

```
Type    tpOcurrenciaNum = record
                            ultimaCifra: integer;
                            ocurrencias: integer
                          end;
        fichOcurrencias = FILE OF tpOcurrenciaNum;
```

b) Desarrollar un segundo programa PASCAL que lea el fichero '*ocurrencias.dat*', obtenido con el programa anterior, y escriba en pantalla cuál es la última cifra con mayor número de ocurrencias.

A continuación, se muestra a modo de ejemplo el fichero que debería generar el programa del apartado (a) y la salida por pantalla producida por el programa del apartado (b) para un fichero concreto de entrada.



Ejercicio 2 [3 puntos]

Sea `n` un número entero positivo de cuatro cifras de modo que no sean todas iguales. Definimos

`may(n)` como el mayor de los números que pueden formarse con esas cuatro cifras,
`men(n)` como el menor de los números que pueden formarse con esas cuatro cifras, y
`dif(n) = may(n) - men(n)`

Por ejemplo, si `n=2111` entonces `may(n)=2111, men(n)=1112` y `dif(n)=0999.`

D. R. Kaprekar descubrió la curiosa propiedad que consiste en que si se repite sucesivamente el mismo cálculo tomando como número `n` para cada iteración el resultado, `dif(n)`, de la anterior, la serie de números obtenidos, `dif(n)`, acaba siempre en el mismo número.

Se pide: Escribir un programa PASCAL que lea de teclado **un número entero** positivo de cuatro cifras y que, una vez que ha comprobado que no todas son iguales, calcule e imprima los valores sucesivos de
`n may(n) men(n) dif(n)` hasta que `dif(n)=n.`

Ejemplo: Para `n=2111`, la salida del programa sería

n	may(n)	men(n)	dif(n)
2111	2111	1112	0999
0999	9990	0999	8991
8991	9981	1899	8082
8082	8820	0288	8532
8532	8532	2358	6174
6174	7641	1467	6174

NOTA:
Dado que las funciones `may(n)` y `men(n)` son similares, basta con incluir la implementación de una de ellas.

Ejercicio 3 [4 puntos]

Sea una secuencia de caracteres introducida por teclado, formada por palabras, y acabada con un punto. Toda palabra está formada exclusivamente por letras del alfabeto (minúsculas), y dos palabras contiguas de la secuencia están separadas por uno o más caracteres distintos de letra.

Dada la siguiente estructura de datos para guardar palabras de hasta 15 caracteres:

```
const  longMaxPalabra  = 15;
type   tpLetrasPalabra = array[1 .. longMaxPalabra] of char;
       tpPalabra = record
                     palabra:     tpLetrasPalabra;
                     longPalabra: 0.. longMaxPalabra
                   end;
```

Se pide:

1) Diseñar la función

   ```
   function esSubpalabra(pal1, pal2: tpPalabra):bolean;
   ```

   que a partir de dos palabras, `pal1` y `pal2`, devuelva *`true`* si y sólo si `pal1` es subpalabra de `pal2` (está contenida).

2) Desarrollar el procedimiento:

   ```
   procedure encontrarPalabras(subpalabra: tpPalabra);
   ```

   que utilizando las declaraciones anteriores y la función *`esSubpalabra`*, determine y escriba por pantalla todas las palabras de la secuencia que contienen la subpalabra especificada como parámetro, `subpalabra` .

   Para facilitar la comprensión del enunciado, a continuación se detalla un ejemplo de ejecución del procedimiento `encontrarPalabras` cuando se especifica como parámetro la subpalabra `'res'`:

   Secuencia de Entrada:

   ```
   los jugadores  mas agresivos se aburrieron; el resultado: goles a pares.
   ```

   Salida por pantalla:

   ```
   jugadores
   agresivos
   resultado
   pares
   ```

Notas: 1) Para simplificar, supondremos que todas las letras de la secuencia son minúsculas de la `'a'` a la `'z'`, y que no hay ninguna palabra con más de 15 letras.

2) No se puede usar el tipo **string**.