Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

20 de junio de 2018

Examen de **Fundamentos de Informática**

Grado en Tecnologías Industriales

Duración total del examen: **3 horas**

NOTA. Responde a cada ejercicio en una hoja diferente. No utilices la misma hoja para dos (o tres) ejercicios.

Ejercicio 1 [3 puntos]

Dado un número entero positivo, su valor *sheldon* es aquél que se obtiene de la siguiente forma: se suman los dígitos que lo componen; si el valor de la suma es menor que el Límite Cooper, el valor sheldon es el valor obtenido; si no, el valor sheldon es el valor sheldon del entero que se ha obtenido en la suma de los dígitos. El Límite Cooper también es un entero positivo, y es mayor que 9.

Se pide desarrollar en Pascal una función que devuelva el valor sheldon de un número, tomando como parámetros dicho número y el valor del Límite Cooper. Si el número no es un entero positivo, la función devolverá -1. La tabla muestra ejemplos de cálculo del valor sheldon para diversos números y valores del Límite Cooper:

Límite Cooper	Número	Cálculo	Valor sheldon
13	7	7	7
	13	$1+3=4$	4
	480	$4+8+0=12$	12
	481	$4+8+1=13;\quad 1+3=4$	4
	16918	$1+6+9+1+8=25;\quad 2+5=7$	7
14	481	$4+8+1=13$	13
12	481	$4+8+1=13;\quad 1+3=4$	4

Nota: No puede emplearse el tipo de dato String, ni vectores, para resolver el ejercicio. Se valorará la eficiencia del algoritmo desarrollado. El programa debe funcionar para enteros arbitrariamente grandes, pero puedes asumir que son representables con el tipo `integer`.

Ejercicio 2 [3.5 puntos]

Una organización guarda en un fichero de texto `contabilidad.txt` todos los asientos contables de su **contabilidad** anual. Cada asiento contable ocupa una línea en el fichero de texto e incluye el código de cuenta del *debe*, el código de cuenta del *haber*, la cantidad del asiento (un número real con dos cifras decimales, en euros) y el concepto (una cadena de texto hasta el fin de línea). Los códigos de cuenta son números naturales de **exactamente tres cifras** y son **únicos**.

Algunas cuentas corresponden a conceptos que la organización no desea que sean de dominio público. Los códigos y conceptos de dichas cuentas están en un fichero de texto `cuentasB.txt`, con el código (3 cifras) y el concepto (una cadena de texto) por cada línea. **Se pide** desarrollar un programa en Pascal que separe los asientos del `contabilidad.txt` en dos ficheros:

- `contabilidadB.txt` que contenga todos los asientos contables que involucren alguna cuenta del fichero `cuentasB.txt` (debe o haber)

- `contabilidadA.txt` donde permanezcan el resto de asientos, de dominio público.

Se recomienda utilizar el tipo de dato **`String`** para representar los conceptos tanto de los asientos contables como de las cuentas.

Ejemplo de ejecución:

contabilidad.txt	**cuentasB.txt**
126 501 11689.15 Pago deuda proveedores 126 721 60000.00 Paga extra en sobre 425 214 121050.00 Venta inmueble 126 845 305000.00 Contrato obra sede 845 215 105000.00 Donacion	721 Sobresueldos 215 Empresa asociada Panama

contabilidadA.txt	**contabilidadB.txt**
126 501 11689.15 Pago deuda proveedores 425 214 121050.00 Venta inmueble 126 845 305000.00 Contrato obra sede	126 721 60000.00 Paga extra en sobre 845 215 105000.00 Donacion

Ejercicio 3 [3.5 puntos]

Un **Tesauro** o Diccionario de Sinónimos es un tipo especial de diccionario que no contiene los significados de las palabras, sin o que estas se agrupan por significados similares o contrarios (sinónimos y antónimos). A esos grupos se les denomina **conceptos**, y nos permiten buscar sinónimos y antónimos de una palabra.

Partiendo de que representamos las palabras mediante el tipo de datos `String` de Pascal, define los tipos y estructuras de datos para represetar el Tesauro:

- `tpGrupoPalabras` que permita representar una secuencia de palabras relacionadas (puede ser un conjunto de sinónimos o de antónimos), con un máximo de 20 palabras.
- `tpConcepto`, que contenga dos grupos de palabras de significados opuestos.
- `tpTesauro`, que contenga un grupo de conceptos hasta un máximo de 1000.

Usando esos tipos de datos, implementa los siguientes procedimientos o funciones:

- Una función que, tomando como entrada una palabra y un grupo de palabras, nos diga si la palabra está en ese grupo:

```
function estaEnGrupo(??? palabra: string;
                     ??? grupo:   tpGrupoPalabras): boolean;
```

- Un procedimiento que, tomando como entrada una palabra y un tesauro, muestre por pantalla los sinónimos y antónimos de dicha palabra, o un mensaje informando de que no existen:

```
procedure muestraAlternativas(??? palabra:  string;
                              ??? tesauro:  tpTesauro);
```

Para comparar palabras puedes usar la siguiente función de Pascal (no tienes que implementarla):

```
function CompareStr(S1: string; S2: string): integer;
```

que devuelve 0 si las dos cadenas son iguales (y otro valor distinto de 0 si no lo son).

Nota: Se valorará que NO haya código repetido, por lo que puedes definir los procedimientos y funciones auxiliares que creas necesario.