Examen de **Fundamentos de Informática**

Grado de Tecnologías Industriales

Escuela de Ingeniería y Arquitectura

Primera convocatoria: 13 de Junio de 2015

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

Ejercicio 1 [3,5 puntos]

El IBEX 35 es el principal índice de referencia de la bolsa española. Está formado por los valores de las 35 empresas con más liquidez que cotizan en las Bolsas Españolas, y fluctúa constantemente.

El IBEX35 puede consultarse en multitud de fuentes, pero todas ellas contienen información similar. En la que vamos a utilizar (ver el ejemplo) figuran, por este orden, el `TKR` o clave del nombre de la compañía, el `Ultimo` valor o precio actual de cada acción, el `RtDiv` o rentabilidad por dividendo (en porcentaje), y el `PER` que es otro índice comparativo.

Un inversor recibe la tabla de cotizaciones del día en un fichero de texto `IBEX35.txt` y decide invertir un pequeño capital comprando acciones de la compañía que tiene la mayor rentabilidad por dividendo (`RtDiv`), En dicho fichero la primera línea son cabeceras y puede desecharse; a continuación, en cada línea y separados por espacios en blanco figuran los datos de cada compañía: `TKR` (texto, 3 o 4 caracteres); `Ultimo` (real), `RtDiv` (real) y `PER` (real).

Fichero **`IBEX35.txt`**

```
TKR Ultimo  RtDiv  PER
ABE 15.935  4.14   19.11
ACS 29.325  3.95   12.84
AMS 41.475  1.56   24.31
ANA 70.130  0.00   23.43
BBVA 8.968  4.13   14.70
BKIA 1.174  0.00   11.62
BKT  6.700  1.95   15.58
BME 37.085  5.10   17.33
CABK 4.363  4.35   16.63
ELE 17.255 82.15   17.30
MAP  3.206  4.37   10.02
OHL 17.180  3.94    7.53
POP  4.551  1.49   21.97
REE 76.670  3.46   16.95
REP 17.280 11.33   15.64
SAN  6.460  9.27   12.26
SCYR 3.915  0.00   15.98
TEF 12.755  5.77   15.94
TL5 11.410  1.14   23.00
TRE 45.585  3.06   17.07
```

En ocasiones las compañías obtienen rentabilidades extraordinarias que no se repiten en el futuro, por lo que se opta por no tener en cuenta las que tienen `RtDiv` > 10%. Así, en el ejemplo, no se elegirán ELE o REP.

Ejemplo de operación:

```
Nombre del fichero: IBEX35.txt
Capital a invertir (eur): 10000
Comprar 1547 acciones de SAN a 6.460
```

Se pide al operador el nombre del fichero y el capital a invertir. A partir de estos datos, se obtienen del fichero los datos de la compañía con mayor `RtDiv (SAN)`, y a continuación el programa calculará el número de acciones a adquirir (`1547`) dividiendo el capital a invertir (`10000`) por el precio de la acción (`6,460`). Finalmente se prepara la orden de compra con el texto indicado en el ejemplo.

Se pide:

1.- Definir las estructuras de datos necesarias para resolver el problema. Se debe usar un registro `tpDato` para guardar en memoria los datos completos de una compañía.

2.- Generar una función que lea un TKR carácter a carácter y lo devuelva en una tabla de tipo `tpTKR=array[1..4] of char;`

```
function leerTKR(??? f:txt):tpTKR;
```
{Pre: el fichero f está ya abierto para lectura y preparado para leer el primer carácter de una línea
Post: Lee del fichero f los caracteres de un TKR, rellenando con un blanco por la derecha si tiene 3}

3.- Desarrollar la función

```
function MaxRtDiv(??? f:txt):tpDato;
```
{Pre: el fichero f está ya abierto para lectura
Post: Devuelve los datos de la compañía con mayor RtDiv en el fichero f}

4.- Finalmente, desarrollar el programa principal `ComprarAcciones` que funcione como explica el ejemplo de operación..

NOTAS: En el fichero **`IBEX35.txt`** del ejemplo se han suprimido los datos de unas cuantas compañías; en el original aparecerían los 35. No hay errores en el fichero.

Ejercicio 2 [3,5 puntos]

Para gestionar la fumigación de un territorio agrícola se subdivide su superficie en sectores de área uniforme. Cada sector se identifica mediante sus coordenadas: un número entre 1 y 5 en el eje norte-sur y un carácter entre la 'A' y la 'E' (siempre en mayúscula y por orden alfabético) para el eje este-oeste. La subdivisión del territorio a fumigar es siempre de 5x5.

Se dispone de un prototipo del software de un dron que permite realizar vuelos autónomos sobre dicho territorio, fumigandolo automáticamente. Al tratarse de un prototipo, los vuelos autónomos pueden no ser perfectos: el dron puede no fumigar ciertos sectores del territorio o puede fumigar varias veces el mismo sector. Tras cada vuelo, el dron guarda el recorrido de dicho vuelo (los sectores fumigados) en un fichero de registros con la siguiente estructura Pascal:

```
tpSector = record
  ns : Integer; {La coordenada en el eje norte-sur.}
  eo : Char;    {La coordenada en el eje este-oeste.}
end;
tpRecorrido = file of tpSector;
```

Se pide:

Desarrollar el procedimiento

```
procedure analizarRecorrido(??? recorrido : tpRecorrido);
{ Pre: el fichero recorrido está ya abierto para lectura
Post: Muestra por pantalla los detalles del recorrido como los indicados en los ejemplos de ejecución}
```

que analiza el vuelo del dron suministrado en `recorrido` e indica por pantalla si el recorrido es perfecto (si ha recorrido todos los sectores sólo una vez) o no. En el caso de que no sea perfecto, el programa mostrará por pantalla una lista con los sectores no fumigados y después otra lista con los sectores que se repiten en el recorrido (incluyendo el número de veces que se repiten). El análisis ignorará los sectores que estén fuera del territorio a fumigar. Define nuevas estructuras de datos si lo consideras necesario.

Ejemplos de ejecución:

EJEMPLO 1	**Fichero**	1A 1B 1C 1D 1E 2A 2B 2C 2D 2E 3A 3B 3C 3D 3E 4A 4B 4C 4D 4E 5A 5B 5C 5D 5E
	Pantalla	Recorrido perfecto
EJEMPLO 2	**Fichero**	3A 3B 3C 3D 3E 3A 3B 3C 3D 3E 3A 3B 3C 3D 3E
	Pantalla	Sectores no fumigados 1A 1B 1C 1D 1E 2A 2B 2C 2D 2E 4A 4B 4C 4D 4E 5A 5B 5C 5D 5E Sectores repetidos 3A (3 veces) 3B (3 veces) 3C (3 veces) 3D (3 veces) 3E (3 veces)
EJEMPLO 3	**Fichero**	0B 0C 0D 1B 1C 1D 2B 2C 2D 3B 3C 3D 4B 4C 4D 5B 5C 5D 6B 6C 6D
	Pantalla	Sectores no fumigados 1A 1E 2A 2E 3A 3E 4A 4E 5A 5E Sectores repetidos

Ejercicio 3 [3 puntos]

Diseñar una función que devuelva el número de veces que aparece una subsecuencia de dos caracteres determinados dentro de una secuencia de caracteres introducida por teclado y terminada con un punto. La función recibe como parámetro la subsecuencia de dos caracteres a buscar. Puede asumirse que tanto la secuencia a analizar como la subsecuencia contienen sólo letras minúsculas.

Diseña también el programa principal que prueba esa función: pide al operador la frase y la subsecuencia a buscar y responde con un texto como el indicado en el ejemplo de ejecución.

```
function apariciones (??? subSec: tpCaracteres): integer;
{Pre: Subsec contiene la subsecuencia a encontrar
Post: Devuelve el número de veces que aparece la subsecuencia subSec de dos caracteres en una secuencia de caracteres introducida por teclado}
```

Ejemplo de ejecución

```
Subsecuencia a buscar: la
Frase: la luz azul aparecia de noche en la llanura
La subsecuencia 'la' aparece 3 veces
```