Examen de **Fundamentos de Informática 16 de Septiembre de 2003.**
Ingenierías Industrial, Química y de Telecomunicación
Dpto. Informática e Ingeniería de Sistemas
Centro Politécnico Superior. Universidad de Zaragoza

Duración total del examen: **3 horas 30 minutos**

Ejercicio 1 [4 puntos]

En la resolución de diversos problemas de ingeniería se utilizan ***matrices cuasivacías***, que son aquellas que contienen un pequeño número de elementos no nulos. Un modo muy simple de describirlas, de forma condensada, consiste en indicar sus dimensiones y, a continuación, todos los valores no nulos junto con su posición relativa. Para numerar las posiciones, supondremos que se comienza en el primer elemento (1,1) y se recorre la matriz por columnas (véase la Fig. 1). Para almacenar de forma permanente la matriz se puede utilizar un fichero de texto.

$$\begin{bmatrix} 3.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3.0 \\ 0 & 0 & 6.7 & 0 & 0 \end{bmatrix} \Leftrightarrow$$

fichero		*significado*
4	5	*matriz de 4 filas y 5 columnas*
1	3.6	*en posición 1 (fila 1, columna 1) valor 3.6*
12	6.7	*en posición 12 (fila 4, columna 3) valor 6.7*
19	-3.0	*en posición 19 (fila 3, columna 5) valor –3.0*

Fig.1 *matriz de números reales de 4×5 y su representación de forma condensada en un fichero de texto*

Se pide desarrollar en PASCAL los siguientes procedimientos (previamente se realizarán todas las declaraciones de los tipos de datos que sean necesarias).

1) Para almacenar y recuperar, respectivamente, matrices de dimensiones cualesquiera (pero menores que 100 × 100) en un fichero de texto:

```
procedure escribirMatrizFtxt(var m: tpMatriz; var f: tpMatrizFtxt);
```
{ almacena, de forma condensada, la matriz m *en el fichero de texto* f *}*

```
procedure leerMatrizFtxt(var m: tpMatriz; var f: tpMatrizFtxt)
```
{ devuelve en m *la matriz que hay almacenada, de forma condensada, en el fichero de texto* f *}*

2) Si se representa cada elemento de la matriz mediante un registro con dos campos: `posicion` y `valor`, se puede utilizar un fichero de registros para almacenar la matriz. En el campo `posicion` del primer registro del fichero se representa el número de filas, y en el campo `valor`, el de columnas. Esta representación permite realizar la suma de matrices trabajando directamente con los ficheros (sin cargarlas en memoria). Obsérvese que, como los elementos están ordenados por posición, la suma es similar a una mezcla de ficheros (Fig.2).

fichero matriz 1			*fichero matriz 2*			*fichero matriz suma*	
4	5.0		4	5.0		4	5.0
1	3.6	+	12	-6.7	=	1	3.6
12	6.7		19	2.0		19	-1.0
19	-3.0						

Fig. 2 *ejemplo de suma de matrices almacenadas en fichero de registros*

```
procedure SumarMatricesFreg(var f1,f2: tpMatrFReg; var fSuma: tpMatrFReg);
```
{ devuelve en `fSuma` *la suma de las matrices (de igual dimensión) almacenadas, de forma condensada, en los ficheros de registros* `f1` *y* `f2`*. Los elementos están ordenados por posición }*

Ejercicio 2 [2,5 puntos]

Una frase se denomina *creciente,* si la sucesión formada por la longitud de las palabras que contiene (según su orden de aparición en la frase) se corresponde con una sucesión creciente de números naturales. Se pide implementar un programa en PASCAL, que lea de la entrada estándar una frase acabada con un punto final, e indique en la salida estándar si la frase es creciente o no. Las frases pueden estar compuestas únicamente por caracteres alfabéticos, espacios en blanco y un punto al final.

Ejemplo:

Entrada	**Salida**
El tio Leo estaba cansado.	La frase es creciente.
Vale.	La frase es creciente.
Esta otra frase no.	La frase no es creciente.

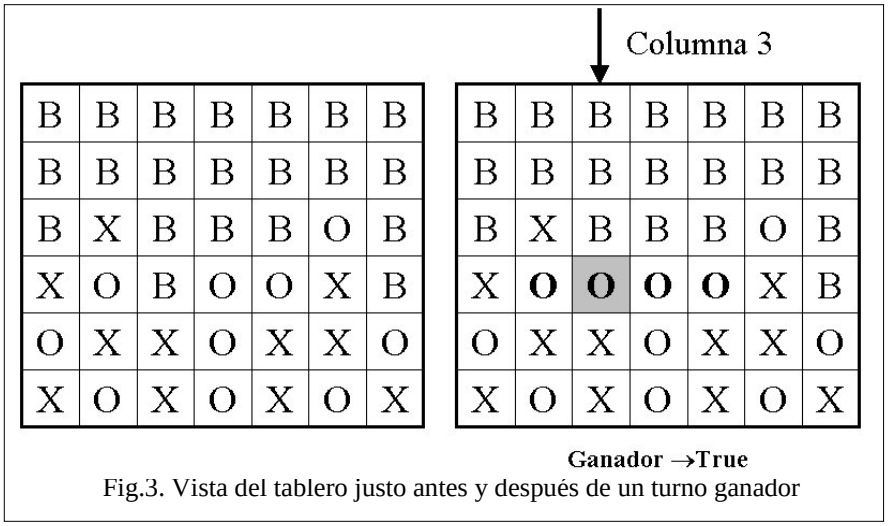
Ejercicio 3 [3,5 puntos]

El Conecta4 es un sencillo juego de estrategia parecido al clásico 4 en raya. El tablero tiene 6 filas y 7 columnas. Juegan dos jugadores por turnos, dejando caer las fichas de una en una, por cualquier columna del tablero. El objetivo final es poner cuatro fichas del mismo color en línea (horizontal, vertical, o diagonal) antes de que lo haga el adversario.

En un momento determinado, cada componente del tablero podrá estar vacía (se representa con la letra B), podrá tener una ficha roja (representada con O), o una ficha verde (representada con X). Antes de iniciarse una nueva partida todas las componentes del tablero están vacías.

Se ha construido parcialmente un programa PASCAL para jugar al Conecta4. Este programa hace jugar de forma alternativa a cada jugador, comenzando siempre cada partida el jugador con fichas rojas. Para la ejecución de cada tirada, el programa solicita al jugador al que le corresponde el turno, que especifique por teclado la columna en la que desea dejar caer su siguiente ficha. Con esta información, se realiza la tirada comprobando: la validez de la misma y si el jugador ha logrado ganar la partida.

Un jugador gana la partida cuando ha logrado cuatro en raya. La última ficha tirada puede estar en cualquiera de las cuatro posiciones.



B	B	B	B	B	B	B
B	B	B	B	B	B	B
B	X	B	B	B	O	B
X	O	B	O	O	X	B
O	X	X	O	X	X	O
X	O	X	O	X	O	X

B	B	B	B	B	B	B
B	B	B	B	B	B	B
B	X	B	B	B	O	B
X	**O**	**O**	**O**	**O**	X	B
O	X	X	O	X	X	O
X	O	X	O	X	O	X

Fig.3. Vista del tablero justo antes y después de un turno ganador

El programa Conecta4 utiliza una función llamada `Ganador` que se encarga de comprobar si una vez realizada una tirada, el jugador ha logrado ganar la partida. Se pide:

1) Completa las estructuras de datos necesarias por el programa Conecta4:

```
const
MaxFilas = 6; MaxColumnas = 7;
type
TpJugador = (jugador1, jugador2); {representaen siguiente jugador en tirar una ficha}
TpCasilla = (B, O, X);            {representa estado de las casillas del tablero}
TpColumna = ¿...?                 {dimensiones del tablero de juego}
TpFila = ¿...?
TpTablero = ¿...?                 {tablero de juego}
TpPartida = record
    tablero : TpTablero;          {almacena estado de la partida}
    turno : TpJugador;            {indica el jugador que le toca tirar la siguiente ficha}
end;
```

2) Utilizando las estructuras de datos previas codifica la función `Ganador` requerida por el programa.

```
function Ganador (¿? p: TpPartida; ¿? c: TpColumna):boolean;
```

{ `c` representa la columna en la que se ha dejado caer la última ficha. Esta función devuelve `true` *si con esa última ficha se ha conseguido tener cuatro en raya (horizontal, vertical, o diagonal). Devuelve* `false` *en caso contrario. En el tablero de la partida* `p` *ya se ha ejecutado previamente la tirada, y por lo tanto, esta función en ningún caso debe modificar su estado.}*