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

Ejercicio 1 [2,5 puntos]

Una **progresión aritmética** de números enteros es una sucesión en la que cada término se obtiene a partir del anterior sumándole un número fijo. Una **progresión geométrica** de números enteros es una sucesión en la que cada término se obtiene multiplicando el anterior por una constante denominada *razón* o *factor* de la progresión.

Se pide: Desarrollar un programa PASCAL que lea del teclado una secuencia de enteros, tan larga como se desee y terminada con "fin de línea", y diga si es una progresión aritmética, geométrica, ambas, o ninguna. Además mostrará el total de números introducidos y, si no se introduce ningún número, o sólo uno, lo indicará de forma oportuna.

Nota: Se puede suponer que tras el último número no hay ningún espacio.

Ejemplos de ejecuciones:

```
1 2 3 4
4 números; forman progresión aritmética
```

```

no hay ningún número
```

```
1 2 4 8 16
5 números; forman progresión geométrica
```

```
7
sólo hay un número
```

```
1 1 1 1 1 1
6 números; forman progresión aritmética
6 números; forman progresión geométrica
```

```
1 2 3 4 7
5 números; NO forman progresión aritm. ni geom.
```

Ejercicio 2 [3,5 puntos]

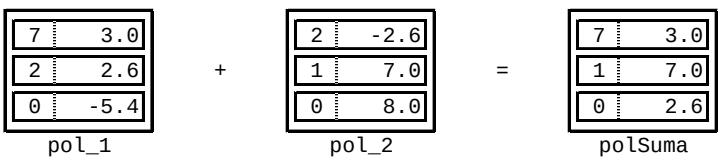
Un polinomio está formado por una secuencia de monomios de distinto grado. Así, el polinomio $3x^7 + 2.6x^2 - 5.4$ está formado por los monomios $3x^7$, $2.6x^2$ y -5.4, de grados 7, 2 y 0, respectivamente. Si se quiere operar con polinomios de grados muy grandes, puede tener interés representar cada polinomio en un fichero de registros, donde cada registro representa un monomio. Sabiendo que en el fichero sólo aparecen representados los monomios no nulos y que éstos están ordenados por grado, de mayor a menor,

Se pide:

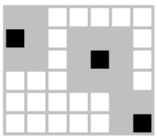
1) Definir los tipos de datos `tpMonomio` y `tpPolinomio` que permiten representar, respectivamente, un monomio y un polinomio.
2) Desarrollar el procedimiento `sumarPolinomios` que, a partir de dos polinomios suministrados a través de los parámetros `pol_1` y `pol_2`, de tipo `tpPolinomio`, devuelva la suma de ambos polinomios a través de un tercer parámetro `polSuma`, también de tipo `tpPolinomio`.

Nota: Dado que el grado de los polinomios puede ser tan grande como se desee, no se permite el empleo de estructuras de tipo vector.

a continuación se muestra la suma de `pol_1` = $3x^7 + 2.6x^2 - 5.4$ y `pol_2` = $-2.6x^2 + 7x + 8$



Ejercicio 3 [4 puntos]



El **juego de la vida** fue diseñado por el matemático británico John Horton Conway en 1970, siendo uno de los mejores ejemplos de un autómata celular.

El tablero de juego es un tablero rectangular que puede tener un tamaño de hasta 40×40 casillas, donde habitan "células". Cada casilla, excepto las de la periferia, tiene ocho casillas vecinas y en cada una de ellas puede, o no, haber una célula; es decir, las casillas tienen dos estados: vacía y ocupada (con una célula). Como ejemplo, en la figura se muestran en gris las casillas vecinas de tres casillas ocupadas (células "vivas", marcadas en negro).

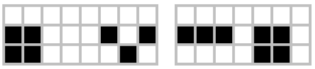
El estado del tablero evoluciona en unidades de tiempo discretas. El estado de todas las casillas en el turno siguiente se calcula siempre a partir del estado actual de las casillas. Puesto que el estado de todas las casillas del tablero se actualiza a la vez, se propone utilizar dos tableros: el actual y el siguiente.

Cada unidad de tiempo se produce un cambio generacional completo, que depende del número de células vecinas que tiene cada casilla, según dos sencillas reglas:

- En una casilla vacía con exactamente 3 células vecinas, "nace" una célula (al turno siguiente la casilla estará ocupada); en caso contrario, permanece vacía.
- Una casilla ocupada que tiene 2 ó 3 células vecinas, sigue ocupada (la célula sigue viva); en caso contrario, la célula muere (por "soledad" o "superpoblación") y en el turno siguiente la casilla estará vacía.

Por ejemplo, si los estados iniciales de estos dos patrones son:

la siguiente generación será:



Se pide:

- Determinar los tipos y estructuras de datos más adecuados para implementar el juego
- Desarrollar los siguientes procedimientos y funciones:

```
procedure iniciarTablero(??? tablero:tpTablero);
{Inicia el tablero a partir de un fichero de texto, cuyo nombre es introducido  por
 el jugador y que tiene el siguiente formato:
 En la primera línea, las dimensiones del tablero (dos enteros en el rango 1..40), y
 en cada una de las siguientes líneas, una secuencia de X y O (X significa casilla con
 célula, y O casilla vacía) que representa una fila del tablero (ver ejemplo)}

function numVecinos(??? tablero:tpTablero;??? fila: tpFila;??? col: tpCol): integer;
{Devuelve el número de células vecinas de la casilla [fila, col] del tablero}

procedure sigGeneracion(??? actual: tpTablero; ??? sigte: tpTablero);
{Calcula el tablero sigte correspondiente a la siguiente generación de actual.
 Puede usarse la función anterior}
```

ejemplo:

- Los estados iniciales de DieHard y Acorn estarían representados en los ficheros `'DieHard.txt'` y `'Acorn.txt'` del siguiente modo:

DieHard.txt
3 8 OOOOOOXO XXOOOOOO OXOOOXXX

Acorn.txt
3 7 OXOOOOO OOOXOOO XXOOXXX