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

Ejercicio 1 [3.5 puntos]

El código genético de las proteínas está formado por una cadena muy larga de bases, representadas por las letras A (*Adenosina*), U (*Uracilo*), C (*Citosina*) y G (*Guanina*). Una terna de estas bases representa un aminoácido, y varios aminoácidos conforman una proteína.

Un aminoácido puede especificarse por varias ternas diferentes (hasta un máximo de 6). Por ejemplo, el aminoácido *Asparagina* se puede especificar por dos ternas: AAU o AAC.

Así, en la cadena genética **ACGAAAUUUACCAUGC**... se podría buscar la *Asparagina* (AAU o AAC) y el resultado sería positivo, hallando la primera base (A) del aminoácido buscado en la posición 5.

Se dispone del fichero de texto **'cod_gen.txt'** que contiene en cada línea la información de una cadena genética. Si sólo se emplean letras mayúsculas, y todas las líneas tienen al menos tres bases, se pide:

1) Diseñar una **estructura de datos** adecuada para representar aminoácidos.
2) Diseñar un **procedimiento** PASCAL que localice en cada cadena genética (línea) del fichero de texto suministrado como parámetro, la primera aparición del aminoácido que se le pasa como segundo parámetro. Debe mostrar por pantalla, para cada línea, la posición inicial donde se ha encontrado el aminoácido (si en una línea no se encuentra dicho aminoácido, se indicará con un 'NO')

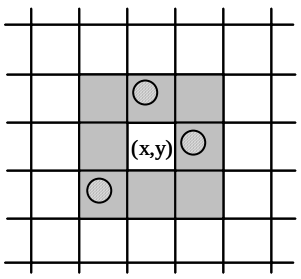
NOTAS: La longitud de las cadenas genéticas puede ser enormemente grande. **No se permite el uso de *strings***. Antes de codificar, se recomienda probar el algoritmo con el ejemplo dado.

Ejercicio 2 [3 puntos]

Se quiere crear un videojuego denominado 'buscaMarcianos', para lo que se dispone de un tablero de **20x20** casillas, en las que se han colocado un número variable de marcianos, distribuidos aleatoriamente. El juego consiste en ir "destapando" casillas hasta encontrar un marciano. Al comienzo de la partida, el jugador desconoce el contenido de todas las casillas. Cuando elige una casilla que no había "destapado", de coordenadas (x,y), su estado pasará de `desconocido` a uno de los dos siguientes:

a) `conMarciano`, si hay marciano en dicha casilla, en cuyo caso se acabará la partida.

b) `sinMarciano`, cuando no hay marciano en la casilla. En este caso, además, se cuenta el número de marcianos que rodean la casilla (un número entre 0 y 8), y se guarda este valor en el tablero para mostrárselo al jugador.



En este ejemplo, el estado de la casilla (x,y) pasará de `desconocido` a `sinMarciano`, y el nº de vecinos pasará a 3, pues existen **3** marcianos en las casillas que la rodean.

¡¡OJO!! Sólo hay que inspeccionar las 8 casillas**, si existen**, que rodean a la posición (x,y) actual (las resaltadas en gris.

Se pide:

1) Definir las **estructuras de datos** necesarias para programar el juego del buscaMarcianos (TpTablero, TpCasilla...) considerando que el estado de una casilla se puede definir con el siguiente tipo enumerado:

```
TpEstadoCasilla = (desconocido, conMarciano, sinMarciano);
```

2) Desarrollar y completar el **procedimiento** que se indica a continuación:

```
PROCEDURE Actualizar_Tablero (??? tablero: TpTablero; ??? posicion: TpPosicion;
                              ??? final_partida: Boolean);
```

{Si el estado de la casilla de `tablero` definida por `posicion` es `desconocido`, el procedimiento devuelve `tablero` con los datos de dicha casilla actualizados: su estado y, si no hay marciano, el número de marcianos vecinos. En `final_partida` indicará si se debe terminar, o no, la partida.}

Ejercicio 3 [3.5 puntos]

Cierto tipo de imágenes (como las que se mandan a través de un fax) están compuestas únicamente por puntos de color blanco y de color negro, sin valores intermedios de gris.

Estas imágenes se pueden representar de manera comprimida en un fichero de enteros **'imagen.cod'**. La codificación empleada en este fichero de enteros consiste en indicar el número de puntos consecutivos que tienen igual valor (blanco o negro), de forma alternativa y comenzando siempre por blanco. Además, el primer dato del fichero representa el número de columnas de la imagen. En el ejemplo de la figura, **'imagen.cod'** codifica una imagen de 12 columnas que tiene cero puntos blancos, cuatro negros, uno blanco, uno negro, tres blancos, etc.

Se pide escribir un programa PASCAL que lea la imagen almacenada de forma comprimida en el fichero de enteros **'imagen.cod'**, y la muestre decodificada sobre el fichero de texto '**imagen_dec.txt'**, utilizando el carácter 'X' para representar los puntos negros y el '.' para los blancos. Además, el programa debe guardar en el fichero de texto '**imagen_cod.txt'** (con el formato que aparece en la figura) la misma información que contiene **'imagen.cod'**, y visualizar en pantalla las dimensiones de la imagen.

A continuación se muestra un ejemplo de codificación en el fichero **'imagen.cod'**, así como los ficheros resultantes del programa pedido, '**imagen_dec.txt'** y '**imagen_cod.txt'**, y la información de las dimensiones de la imagen.

NOTA: se puede suponer que en el fichero **'imagen.cod'** no hay errores de codificación.