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

Ejercicio 1 [3 puntos]

En el fichero de registros '*cuadrilateros.dat*' se almacena información para construir cuadriláteros convexos en el cuadrante positivo del plano X-Y y cuya base sea paralela al eje X. La información de cada cuadrilátero está compuesta por las coordenadas X e Y de cada uno de los cuatro vértices del mismo. Los vértices de cada cuadrilátero siempre se almacenan en el mismo orden empezando por el vértice superior izquierdo y siguiendo con los vértices sucesivos en el sentido de las agujas del reloj.

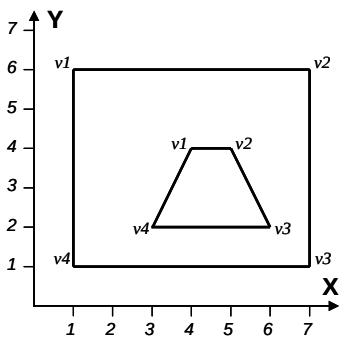
Los tipos PASCAL de los registros y del fichero son los siguientes:

```
Type    tpVertice = record
                      x, y: real  {Coordenadas X,Y del vértice}
                    end;
        tpCuadrilatero = record
                           v1, v2, v3, v4: tpVertice; {vértices del cuadrilátero}
                         end;
        tpFichCuadrilateros = FILE OF tpCuadrilatero;
```

Se pide:

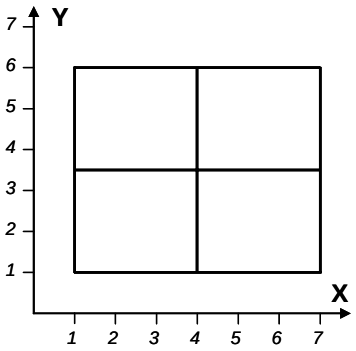
Desarrollar un programa PASCAL que lea el fichero '*cuadrilateros.dat*' y, para cada cuadrilátero leído, determine si se trata de un rectángulo o no. Si el cuadrilátero leído es un rectángulo, se escribirá en el fichero de tipo `tpFichCuadrilateros`, '*rectangulos.dat*', la información correspondiente a los cuatro rectángulos (iguales) que se obtienen uniendo los puntos medios de los lados del rectángulo original leído.

A continuación, se muestra un ejemplo de los ficheros de entrada y de salida y sus correspondientes interpretaciones en el plano X-Y.



1.0	6.0
7.0	6.0
7.0	1.0
1.0	1.0
4.0	4.0
5.0	4.0
6.0	2.0
3.0	2.0

'cuadrilateros.dat'



1.0	6.0
4.0	6.0
4.0	3.5
1.0	3.5
4.0	6.0
7.0	6.0
7.0	3.5
4.0	3.5
4.0	3.5
7.0	3.5
7.0	1.0
4.0	1.0
1.0	3.5
4.0	3.5
4.0	1.0
1.0	1.0

'rectangulos.dat'

Ejercicio 2 [3 puntos]

Un *tramo ordenado alfabéticamente* (TOA) es una secuencia ordenada de letras minúsculas $c_1\ c_2\ c_3 \ldots\ c_n$ (donde cada c_i es un dato de tipo carácter), tal que $c_1 \le c_2 \le c_3 \le \ldots \le c_n$. (orden alfabético)

Se pide: Desarrollar un programa PASCAL, *ContadorDeTramos*, que lea del teclado una secuencia de caracteres, formada exclusivamente por letras minúsculas y finalizada con un punto, y calcule los tramos TOA contenidos en dicha secuencia. De cada tramo será necesario calcular su longitud (es decir, el número de letras del tramo) y su primera y última letra. El resultado de la ejecución del programa será almacenado en el fichero de texto '*tramos.txt*', que contendrá en cada línea la información de un tramo, es decir, su longitud, primera letra y última letra, separados entre sí por un espacio en blanco.

Por ejemplo, dada la siguiente secuencia de entrada: **`bacfmnrgjlr.`**
el programa debería identificar los tramos:
`'b'`, **`'acfmnr'`**, y **`'gjlr'`** y generar como resultado un fichero con el siguiente contenido:

```
1 b b
6 a r
4 g r
```

Ejercicio 3 [4 puntos]

Diseñar un programa PASCAL que permita a un usuario jugar contra el computador al juego del ahorcado. En este juego, el jugador tiene que adivinar una palabra secreta generada por el computador, proponiendo una letra cada vez que juega. El juego se desarrolla en una secuencia de jugadas, cada una de las cuales se compone de las siguientes operaciones:

1) Mostrar en pantalla la palabra adivinada hasta ese momento; es decir, se muestra la palabra secreta, pero sustituyendo las letras no adivinadas por un guión '-', junto a las vidas que le quedan.
2) Lectura de la letra propuesta por el usuario.
3) Si la letra introducida pertenece a la palabra secreta, en la siguiente jugada aparecerá en las posiciones correctas. En caso contrario, se habrá producido un fallo que restará una vida al jugador. Al comienzo del juego el usuario dispone de 7 vidas (fallos permitidos).

El juego termina cuando se ha adivinado la palabra (el usuario gana) o se produce un fallo cuando quedan 0 vidas (el usuario pierde y se le muestra la palabra secreta que tenía que adivinar).

A continuación se especifican las estructuras de datos que se utilizarán en el diseño del programa, junto con un ejemplo de ejecución del juego, tal y como aparece en pantalla. Para seleccionar la palabra secreta con la que se inicia el juego, se dispone del procedimiento `generarPalabra` que, a través del parámetro `palabra`, devuelve la palabra (en mayúsculas) que hay que adivinar.

```
const   MAXLONG=20;   {longitud máxima de una palabra}
        MAX_VIDAS=7;  {vidas disponibles al comienzo del juego}
type
  tpPalabra = record
                long: 0..MAXLONG; {longitud de la palabra}
                vChar: array [1..MAXLONG] of char
              end;
procedure generarPalabra(var palabra: tpPalabra);
{ devuelve a través de palabra la palabra que hay que adivinar }
```

```
La palabra seleccionada es:
-----     vidas: 7
Introduce una letra: A
-----     vidas: 6
Introduce una letra: E
-----     vidas: 5
Introduce una letra: I
----I     vidas: 5
Introduce una letra: O
----I     vidas: 4
        . . . .
      (el juego continúa…)
        . . . .
--U-I     vidas: 1
Introduce una letra: S
--U-I     vidas: 0
Introduce una letra: M
Lo siento, has perdido.
La palabra era:  FLUVI
```

```
La palabra seleccionada es:
-------     vidas: 7
Introduce una letra: A
A----A-     vidas: 7
Introduce una letra: E
A----A-     vidas: 6
Introduce una letra: P
AP---A-     vidas: 6
  . . . .
      (el juego continúa…)
  . . . .
APR-BAR     vidas: 2
Introduce una letra: O
APROBAR     vidas: 2
Enhorabuena, has ganado!
```

Nota: Se supone que el usuario introduce siempre caracteres válidos (letras mayúsculas y vocales sin acentuar)