Examen de **Fundamentos de Informática**
Grado de Tecnologías Industriales
Escuela de Ingeniería y Arquitectura
Segunda convocatoria: 4 de Septiembre de 2015

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

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

Ejercicio 1 [3 puntos]

En matemáticas un número *n-Harshad* es un entero que, escrito en base *n*, es divisible por la suma de sus dígitos (operando en base *n*). Estos números fueron definidos por D. R. Kaprekar y la palabra *Harshad* proviene del sánscrito y significa "gran alegría". Asumiendo que la base es 10, tendremos:

Ejemplos:

- El número 18 es un número *10-Harshad* (número *Harshad* en base 10), ya que la suma de sus dígitos es 9 (1+8=9), y 18 es divisible por 9.
- El número 19 no es un número *10-Harshad*, ya que la suma de sus dígitos es 10 (1+9=10), y 19 no es divisible por 10.
- El número 1729 es un número *10-Harshad*, ya que la suma de sus dígitos es 19, y 1729=19*91.
- Los primeros números *10-Harshad* mayores que la base son:

  12, 18, 20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50, 54, 60, 63, 70, 72, 80

Se pide:

Desarrollar un programa PASCAL que muestre por pantalla la sucesión de números *10-Harshad* comprendidos entre dos números a y b (a<=b y a>0) introducidos interactivamente por el usuario.

El último ejemplo muestra la salida del programa con a=12 y b=80.

Ejercicio 2 [3,5 puntos]

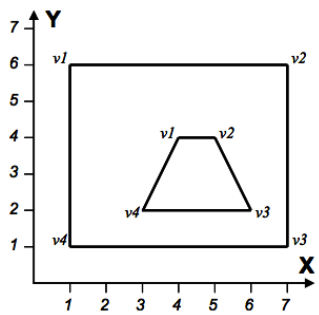
El fichero de registros *cuadrilateros.dat* almacena información para construir cuadriláteros convexos dentro del cuadrante positivo del plano *X-Y* y cuya base es paralela al eje *X*. La información de cada cuadrilátero está compuesta por las coordenadas *x*, *y* de cada uno de sus vértices, comenzando por el vértice superior izquierdo y continuando en el sentido de las agujas del reloj (véase el ejemplo).

Se pide:

Desarrollar un programa PASCAL que lea el fichero *cuadrilateros.dat* y para cada cuadrilátero leído determine si es, o no, un rectángulo. En caso afirmativo, escribirá en el fichero *rectangulos.dat* los datos correspondientes a los cuatro sub-rectángulos resultantes de unir los puntos medios de los lados del rectángulo original leído. A continuación se muestra un ejemplo de fichero de entrada (con su interpretación geométrica), así como el fichero generado por el programa pedido (también con su interpretación geométrica).

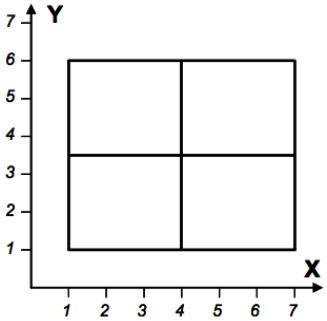
Las estructuras de datos a utilizar son:

```
tpVertice = record
              x,y: real
            end;
tpCuadrilatero =  record
                    v1, v2, v3, v4: tpVertice
                  end;
tpFichCuadrilateros = FILE OF tpCuadrilatero;
```



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 3 [3,5 puntos]

Una **imagen digital** no es más que una matriz bidimensional de elementos llamados píxeles, cuyos valores numéricos indican el color de la correspondiente región de la foto. En el caso de una imagen en **escala de grises**, estos valores numéricos van desde 0 (que indica negro) hasta 255 (que indica blanco). Los valores intermedios representan, por tanto, distintas gradaciones de gris.

Existen numerosas técnicas de tratamiento digital que se basan en alterar los datos de una imagen para obtener distintos efectos. Así, por ejemplo, el ***filtro de caja (o de media)*** emborrona la imagen, como se puede apreciar en la Figura 1. Para el cálculo de un filtro simple de caja, se sustituye el valor de cada pixel por la **media de los valores de dicho píxel y sus vecinos** (píxeles de arriba, abajo, izquierda, derecha, y los ubicados contiguamente en las diagonales). Es importante resaltar que un píxel puede tener 8, 5 ó 3 vecinos, dependiendo de su posición en la imagen.

Se pide:

1) Definir la estructura de datos `tpImagen`, que permita guardar una imagen en memoria para su tratamiento posterior. Se considerará que una imagen podrá tener un tamaño cualquiera de como máximo 1000x1000 píxeles.
2) Desarrollar el siguiente procedimiento:

```
procedure filtroCaja(??? original: tpImagen;  ??? filtrada: tpImagen);
```
{ Aplica un filtro de caja a la imagen `original`, guardando el resultado en la imagen `filtrada` }

a) Imagen original

b) Imagen filtrada

Figura 1: Ejemplo

```
[   1    4  211   13]          [  88   94   90   71]
[  93  255    5   56]          [  91   84   97   81]
[  67  128    0  202]          [ 135  91  107   65]
```

a) Valores imagen original *b) Valores imagen filtrada*

Filtrada[1,1] = (1+4+93+255) div 4 = 88
Filtrada[3,2] = (4+211+13+255+5+56+128+0+202) div 9 = 97
Filtrada[2,3] = (93+255+5+67+128+0) div 6 = 91

c) Ejemplos de cálculo de píxeles de la imagen filtrada

Figura 2: Cálculo del filtro de media