Ejercicio 1 [3.5 puntos]

Dada una secuencia de caracteres terminada con un '.' (punto) y formada por palabras separadas por uno o varios espacios en blanco, se pide escribir el procedimiento

```
procedure editarPaginas(maxCarLinea, maxLineasPagina: integer);
```

que lee una secuencia de la entrada estándar y escribe en pantalla el mismo texto de acuerdo al siguiente formato de líneas y páginas:

- cada línea tiene como máximo *maxCarLinea* caracteres, y
- cada página tiene como máximo *maxLineasPagina* líneas.

Durante el proceso de escritura, las palabras no se pueden partir, es decir, si una palabra completa no puede escribirse en la línea actual, deberá escribirse en una nueva línea. Por simplicidad, supondremos que no existen palabras más grandes que la longitud de una línea y que la longitud máxima de una palabra es de 15 caracteres.

Además, para cada página, el programa debe imprimir una cabecera de página con el número de página en la parte superior, tal como se indica en el ejemplo presentado a continuación.

Ejemplo: Sea la secuencia de entrada,

```
Aun**no*me*he*inscrito***aquí*estaban*muy*liados*y*preferi*dejarlo*para*mas*adelante.
```

Por claridad, cada espacio en blanco se representa en la secuencia del ejemplo como un asterisco. Dada esta secuencia concreta, el resultado correcto de invocar al procedimiento *editarPaginas(20,4)* es:

```
--------
Pagina 1
--------
Aun**no*me*he*
inscrito***aquí*
estaban*muy*liados*y
*preferi*dejarlo*
--------
Pagina 2
--------
para*mas*adelante
```

Nota: En la solución del ejercicio no podrá utilizarse el tipo STRING.

Ejercicio 2 [3 puntos]

Se necesita disponer de una operación especial de suma de números naturales que denominaremos `sumaOEX`. Dados dos números naturales `a` y `b`, `sumaOEX(a,b)` es el número natural correspondiente a la suma bit a bit y sin acarreo de los números `a` y `b`, como se ilustra en el siguiente ejemplo:

```
  0101   ≡ 5
+ 0011   ≡ 3
  0110   ≡ 6
sumaOEX(5,3)=6
```

Se pide escribir la siguiente función Pascal:

```
function sumaOEX (a, b:integer): integer;
{Devuelve la suma bit a bit y sin acarreo de los números naturales a y b}
```

SUGERENCIA (aunque existen otras muchas soluciones válidas)

No es necesario almacenar la representación de los números en binario en ninguna estructura de datos para luego realizar la suma. Se pueden ir obteniendo los dígitos (0,1) de los correspondientes números en binario a la vez, operar con ellos, e ir construyendo el número resultante en base 10.

A continuación se presenta un "dibujo" que muestra cómo se puede obtener un número en binario, comenzando primero por las cifras menos significativas.

```
11 |2
 1  5 |2
    1  2 |2
       0  1 |2
          1  (0)
```

$1011_{base2} \equiv 11_{base10}$

Ejemplo para pasar un número en base dos a base 10:

$$0110_{base2} \equiv 6_{base10} \equiv 2^3 \times \mathbf{0} + 2^2 \times \mathbf{1} + 2^1 \times \mathbf{1} + 2^0 \times \mathbf{0}$$

Ejercicio 3 [3.5 puntos]

Recientemente, el Gobierno de Aragón ha realizado un concurso-oposición para la provisión de 30 plazas de profesor de primaria. La información referente a las plazas ofertadas está almacenada en un fichero de registros de tipo `tpPlaza`, llamado `'plazas.dat'`. El tipo `tpPlaza` consta de los siguientes campos: `idPlaza`, identificador único de la plaza, e `idCentro`, identificador único del centro.

```
type  tpPlaza =  record
                   idPlaza: integer;
                   idCentro: integer
                 end;
      tpFicheroPlaza = FILE OF tpPlaza;
```

Por otro lado, los opositores que han superado el concurso han rellenado una solicitud indicando sus preferencias respecto a las plazas ofertadas. Todas las solicitudes han sido procesadas y almacenadas en un fichero de texto llamado `'solicitudes.txt'`. El formato de este fichero es el siguiente:

- Cada línea tiene la información de un opositor. Más concretamente, consta de 31 números enteros (menores que `maxInt`) separados por espacios en blanco, donde: el primer número representa el DNI del opositor y los 30 siguientes los identificadores de plaza ordenados según sus preferencias (el primero es el más preferente y el último el menos preferente). Por simplicidad, podemos suponer que en la información de un opositor no hay números de plaza repetidos, ni números de plaza inexistentes con respecto los identificadores de plazas ofertadas.

- Las solicitudes, representadas por cada línea, están ordenadas por la nota que han obtenido los opositores. De esta forma, la primera línea del fichero contiene la información del opositor que obtuvo la nota más alta del concurso, la segunda línea la información del que obtuvo la segunda mejor nota, y así sucesivamente.

- Puede haber más opositores que hayan superado el concurso que plazas ofertadas. No obstante, sólo los 30 con mejor nota podrán obtener una plaza de acuerdo a sus preferencias.

Se pide construir un programa PASCAL que implemente el proceso de adjudicación de plazas. Las plazas serán asignadas por nota, de mayor a menor. Además, a cada opositor se le adjudicará una única plaza conforme sus preferencias. Más concretamente, se le concederá la primera plaza que aún esté libre (es decir, no haya sido asignada previamente a otro opositor que obtuvo mejor nota) en su lista de preferencias. El resultado de la adjudicación deberá ser almacenado en un fichero de texto, llamado `'resultado.txt'`, donde en cada línea se guarda el DNI del opositor y el identificador de la plaza que se le ha adjudicado. En el fichero resultado, los opositores también deberán estar ordenados por nota (es decir, en el mismo orden que en el fichero de solicitudes).

NOTAS:

1) Se podrán definir las estructuras de datos auxiliares que le ayuden a resolver el problema
2) El programa deberá funcionar correctamente si el número de plazas a adjudicar es menor que 30