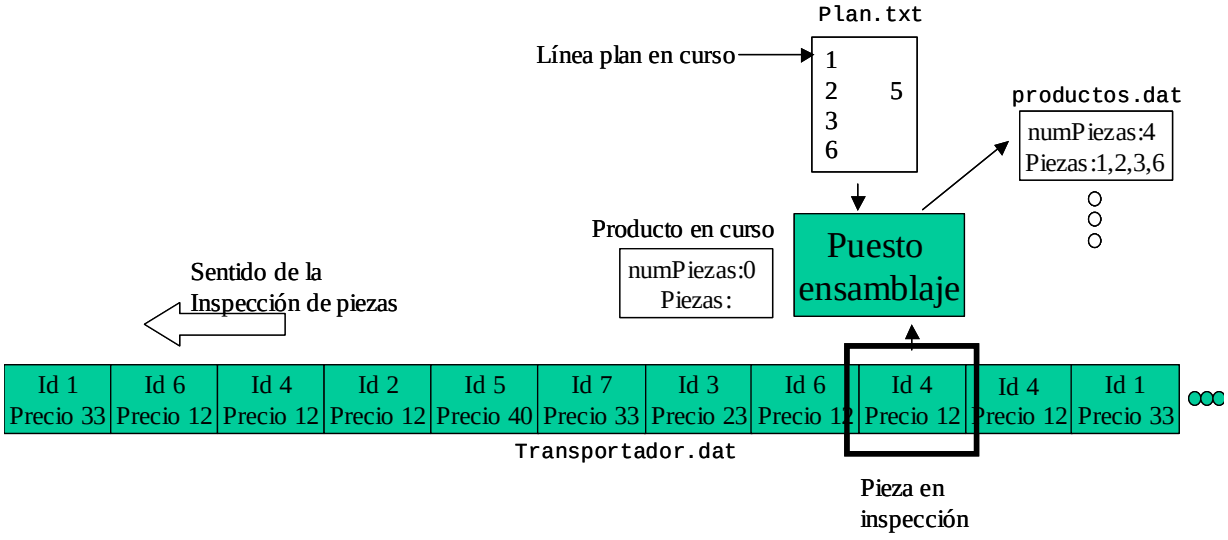
Duración total del examen: 3 horas

Ejercicio 1 [4 puntos]

Se va a realizar un sencillo simulador de un puesto de trabajo en un sistema de fabricación. Supondremos que las piezas entran por una cinta transportadora y pasan por delante del puesto de trabajo donde se ensamblan. Cada puesto ensambla un tipo de **producto** (de tipo `tpProducto`) de acuerdo a un plan y es capaz de coger de la cinta las piezas que necesita ensamblar cuando pasan por delante del puesto. Como máximo un producto tendrá 6 piezas a ensamblar.

Tenemos un fichero que simula la cinta transportadora, `'transportador.dat'`, que contiene las piezas de `tpPieza`, en el orden por el que pasan por delante del puesto. Supondremos un único puesto por simplificar.

El objetivo del puesto es ensamblar todas las unidades posibles del producto de acuerdo al plan. El plan está en un fichero de texto `'Plan.txt'`, en el que aparecen en cada línea los tipos de pieza identificados por un entero (ID). Cuando para el ensamblaje sea posible mas de un tipo de pieza aparecen en la línea los tipos de pieza posibles. Como máximo aparecen tres tipos de pieza alternativos por línea, y como mínimo uno. Para una mejor comprensión, en el ejemplo siguiente se muestra el estado de la simulación una vez completado el ensamblaje de un producto de acuerdo al plan.



Se pide un **Programa pascal** que simule el proceso de fabricación del puesto de trabajo. Se comenzará por la primera línea del plan. En cada paso de simulación se leerá (**mirará**[1]) una pieza del fichero `'transportador.dat'`. Si el ID de la pieza corresponde con alguno de los que se pueden utilizar de acuerdo al plan, se utilizará la pieza para el producto y pasaremos a la siguiente línea del plan. Cuando se tengan suficientes piezas (un plan de trabajo finalizado) se guardará el producto ensamblado en el fichero de `tpProducto` `'productos.dat'`, y comenzaremos el ensamblado de un nuevo producto conforme al

[1] **Se entiende por mirar, inspeccionar la pieza, no se pide eliminarla del fichero.**

plan. El programa finalizará cuando no queden más piezas por procesar en el fichero `'transportador.dat'`.

Puedes utilizar los siguiente tipos.

```
CONST
MAXPIEZAS = 6;    {6 piezas como máximo en un producto}

TYPE
tpPieza = record
             id: integer;   {identificador clase de pieza}
             precio: real;  {precio en €}
            end;

tpProducto = record          {Agrupa las piezas que forman el producto}
             piezas: array[1..MAXPIEZAS] of tpPieza;
             numPiezas: 0..MAXPIEZAS;
            end;
```

Puedes proponer tipos adicionales si lo estimas oportuno. Los tipos `tpPieza` y `tpProducto` no se pueden alterar puesto que debes leer un fichero de `tpPiezas` y crear un fichero de `tpProducto`.

Ejercicio 3 [2,5 puntos]

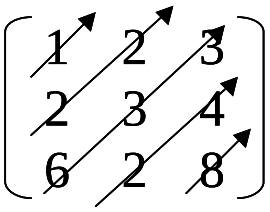
Dado el tipo `tpMatriz`, se pide escribir el procedimiento Pascal `sumaDiagonales` que escriba por pantalla el resultado de la suma de cada una de las diagonales de una matriz cuadrada. El tratamiento de las diagonales deberá realizarse conforme al orden y sentido descrito en el ejemplo.

```
CONST
      MAX = 100;

TYPE
      tpIndice = 1..MAX;
      tpMatriz = record
                   dim: tpIndice;
                   valores: array[tpIndice, tpIndice] of integer;
                 end;

procedure sumaDiagonales (¿? m: tpMatriz);
{Escribe por pantalla el resultado de la suma de cada una de las diagonales de m}
```

Ejemplo:

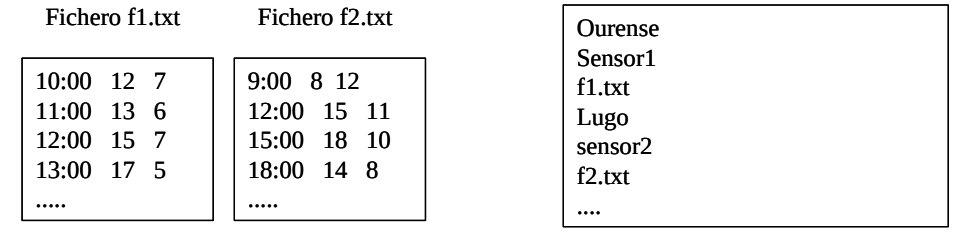



```
El resultado es:1  4   12   6   8
```

Duración total del examen: 3 horas

Ejercicio 2 [3,5 puntos]

El creciente número de incendios en Galicia ha llevado a las autoridades competentes a instalar una colección de sensores físicos en lugares considerados como de máximo riesgo. Cada sensor mide con una frecuencia determinada la temperatura en grados del ambiente y la velocidad del viento en m/seg. Al final de cada día, se almacenan sus mediciones en un fichero de texto (véase ejemplo en la parte izquierda de la figura) donde por cada línea se escribe la hora de la medición (en formato hh:mm) y la temperatura y la velocidad del viento en ese momento del día (ambas representadas como enteros).



Ejemplos de ficheros de medición de sensores

Ejemplo de fichero 'sensores.txt'

Por otro lado, la ubicación de los sensores se guarda en el fichero de localizaciones `'sensores.txt'`. Por cada sensor se almacenan en tres líneas consecutivas el nombre de la provincia donde está instalado, su identificador y el nombre del fichero que contiene sus últimas mediciones (parte derecha de la figura). Evidentemente, en una provincia puede haber instalados más de un sensor.

El objetivo final es detectar alertas de incendio. Una alerta se produce cuando en una medición la temperatura y la velocidad del viento son mayores que unos valores máximos permitidos. Para su posterior análisis, se necesita registrar por cada alerta el identificador del sensor que la detectó, la hora en la que se produjo, y la temperatura y velocidad del viento medidos. La experiencia demuestra que nunca se producen más de 100 alertas diarias.

Se pide: (1) declarar las estructuras de datos necesarias (como mínimo son necesarias las indicadas a continuación); y (2) diseñar en Pascal los procedimientos `leerProvincia` y `deteccionAlertas`:

```
TYPE
    tpInstalacionSensores = text;
    tpInfoSensor = text;
    tpProvincia  = (Ourense, Acoruna, Lugo, Pontevedra);
    tpAlerta = ....
    tpAlertasDiarias = ...

procedure leerProvincia (¿? f: text; ¿? provincia: tpProvincia);

{Lee del fichero de texto f una cadena que identifica la provincia y lo devuelve a
través del parámetro de tipo tpProvincia}
```

```
procedure deteccionAlertas (¿? f: tpInstalacionSensores;
                            ¿? provincia: tpProvincia; ¿? tempMaxima: integer;
                            ¿? velMax: interger; ¿? alertas: tpAlertasDiarias);

{Dada una provincia, unos valores máximos de temperatura y velocidad del viento, y
el fichero f donde se describe la ubicación de los sensores y los ficheros con las
mediciones, detecta todas las posibles alertas de incendio que tienen lugar en esa
provincia y las devuelve en el parámetro alertas}
```

Importante: **En la corrección del examen se valorará muy positivamente la eficiencia, legibilidad y modularidad de las soluciones diseñadas.**