Duración total del examen: **3 horas 30 minutos**

Ejercicio 1 [4 puntos]

El objetivo de este programa es generar listados de pruebas olímpicas de carrera (natación, atletismo, etc). Estos serán de dos tipos: los **listados previos** a las pruebas, donde aparecen por orden de calle de salida la calle, dorsal, nombre y código de país de cada participante, y los **listados finales** de las pruebas, donde aparecen el puesto obtenido, tiempo, dorsal, código de país y nombre de cada atleta, ordenados por tiempos.

Se parte de la información contenida en tres ficheros de texto. El primero (*Participantes.txt*) contiene los datos de todos los participantes que se han inscrito inicialmente, y dedica tres líneas a cada participante: dorsal (entero), nombre (cadena de 50 caracteres máx.) y el código de su nación (3 letras).

El segundo fichero (*OrdenDeSalida.txt*) contiene en una sola línea los dorsales de los participantes en la prueba concreta a celebrar (semifinal, final,...) separados por espacios en blanco y ordenados según el número de calle por la que correrá cada uno. Supondremos que la prueba admite ocho atletas como máximo; en su caso, las calles no ocupadas son las últimas.

El tercer fichero (*Tiempos.txt*) se genera inmediatamente después de la prueba, y contiene tantas líneas como participantes. En cada una de ellas aparecen, por orden de número de calle, los tiempos obtenidos por cada deportista en el formato `mm:ss:cc`. Por ejemplo, `05:23:45` significa 5 min 23,45 s. Si algún participante es descalificado o no acaba la prueba, su tiempo aparece como `59:59:99`

Se pide:

1) Definir las estructuras de datos más adecuadas para resolver el problema.
2) Diseñar un programa PASCAL que presente por pantalla el **listado previo** y el **listado final** de la prueba con los formatos indicados en el ejemplo, a partir de los datos de los tres ficheros de texto.

Ejemplo: Carrera de 100 m lisos, Atenas 2004. Los ficheros de datos son :

Fichero 1: Participantes.txt *(contiene todos los inscritos en la fase previa)*

1763	1965	3003
POGNON Ronald	ZAKARI Aziz	COLLINS Kim
FRA	GHA	SKN
2229	2596	2241
FRATER Michael	EMEDOLU Uchenna	POWELL Asafa
JAM	NGR	JAM
3246	1136	2739
CRAWFORD Shawn	THOMPSON Obadele	OBIKWELU Francis
USA	BAR	POR
3263	1959	3268
GATLIN Justin	MYLES-MILLS Leonard	GREENE Maurice
USA	GHA	USA
1815	1235	2251
LEWIS-FRANCIS Mark	LIMA Vicente	THOMAS Dwight
GBR	BRA	JAM

(Sigue a la dcha.) *(Sigue a la dcha.)*

Fichero 3: Tiempos.txt
(de la final)

00:10:00
59:59:99
00:09:85
00:09:89
00:09:86
00:09:94
00:09:87
00:10:10

Fichero 2: OrdenDeSalida.txt
(en la final; trabajaremos sobre esta prueba)

3003 1965 3263 3246 2739 2241 3268 1136

Listados que se piden

Listado previo

```
**** PARTICIPANTES ****

 1  3003   SKN  COLLINS Kim
 2  1965   GHA  ZAKARI Aziz
 3  3263   USA  GATLIN Justin
 4  3246   USA  CRAWFORD Shawn
 5  2739   POR  OBIKWELU Francis
 6  2241   JAM  POWELL Asafa
 7  3268   USA  GREENE Maurice
 8  1136   BAR  THOMPSON Obadele
```

Listado final

```
   **** RESULTADOS FINALES ****

 1   0: 9:85  3263   USA  GATLIN Justin
 2   0: 9:86  2739   POR  OBIKWELU Francis
 3   0: 9:87  3268   USA  GREENE Maurice
 4   0: 9:89  3246   USA  CRAWFORD Shawn
 5   0: 9:94  2241   JAM  POWELL Asafa
 6   0:10: 0  3003   SKN  COLLINS Kim
 7   0:10:10  1136   BAR  THOMPSON Obadele
 8  No Calif  1965   GHA  ZAKARI Aziz
```

Ejercicio 2 [3 puntos]

Para representar cadenas de caracteres, se proponen las siguientes definiciones:

```
const maxLongCad = 80;
type  tpLongCad = 0..maxLongCad;
      tpCad80 = record
                  long:  tpLongCad;
                  vChar: array [1..maxLongCad] of char
                end;
```

Se pide desarrollar en PASCAL la siguiente función:

```
function  posicion (c1, c2: tpCad80): tpLongCad;
```

{ Si la cadena `c1` está incluída en la cadena `c2`, devuelve la posición del primer carácter coincidente; en caso contrario, devuelve 0 }

Ejemplos: la posición de `'la'` *en* `'hola'` *es 3; la posición de* `'mola'` *en* `'hola'` *es 0;*

Ejercicio 3 [3 puntos]

Para representar conjuntos de letras (minúsculas) del alfabeto, se proponen las siguientes definiciones:

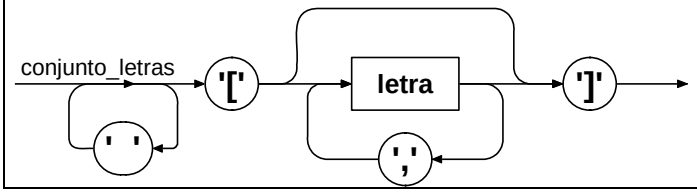
```
type  tpLetra = 'a'..'z';
      tpConjLetras = array [tpLetra] of boolean;
```

De esta forma, una variable `c` de tipo `tpConjLetras` representa a un conjunto C de letras (colección de letras no repetidas); por ejemplo, en la representación del conjunto `['a','d','h','n']` `c['a']= c['d']= c['h']= c['n']=true`, y el resto `c['b']=. . .=false.`

Se pide desarrollar en PASCAL los siguientes algoritmos:

`function incluido (c1, c2 : tpConjLetras): boolean;` *{ devuelve* `c1` $\subseteq$ `c2` *}*
{ por ejemplo, si `c1=['d','h']` *y* `c2=['a','d','h','n']` *entonces* `incluido(c1, c2) = true`*}*

`procedure leerConjLetras (var c : tpConjLetras);`
{Lee del teclado un conjunto de letras y lo devuelve en `c`*. La sintaxis que seguirá el usuario para la introducción de un conjunto es la siguiente:*



Ejemplos:
`[]` *es el conjunto vacío;*
`[a,z]` *es el conjunto formado por* `'a' y 'z';`
`[z,d,a,b]` *es el conjunto formado por* `'a', 'b','d' y 'z'.`

Se supondrá que no se cometen errores en la introducción de la secuencia de caracteres}