I.T. Informática Gestión y Sistemas – Programación I

Examen convocatoria ordinaria. 2 de febrero de 2010.

Apellidos _____

Nombre _____ **Grupo** _____

DNI y Firma

- Duración del examen: 2 horas y media.
- Resolver los problemas en una página diferente cada uno de ellos.
- Poner nombre y apellidos en todas las páginas.
- Las hojas del enunciado también deben entregarse.
- Se valorará la presentación y la claridad en la exposición.
- Se valorará la adecuación de las estructuras utilizadas al problema a resolver.
- No se calificarán las respuestas escritas a lápiz.

1. (3 *puntos*) En una determinada empresa los directivos deben tener una reunión de control (de **una hora** de duración) cada semana, en un día y hora que sean compatibles para todos ellos. Cada directivo tiene asociado una mini-agenda semanal en la que indica cuales son las horas disponibles para la siguiente semana, y con la información de todas estas agendas se busca el día y hora en la que todos los directivos están disponibles para llevar a cabo la reunión. El objetivo de este problema es crear un programa que lleve a cabo esta tarea.

El horario de trabajo de la empresa es de lunes a viernes en el rango horario **HorIni..HorFin**. El formato de la agenda de cada directivo esta descrito por el tipo de datos **TAgenda:** Para cada día y hora existe un valor lógico que indica si esa hora está disponible o no para la reunión (**Nota**: En esta empresa todas las tareas se programan a las horas en punto y ocupan un número discreto de horas). Se dispone de un fichero, **`'agendas.dat'`**, de tipo **TFichAgendas** que almacena las agendas de todos los directivos para la semana en que se quiere celebrar la reunión.

El programa tomará como datos de entrada las agendas contenidas en el fichero, y deberá buscar un día de la semana y una hora donde todos los directivos estén disponibles. Si hay varias alternativas, se escoge el **primer** día posible, y dentro de ese día la hora **más temprana**.

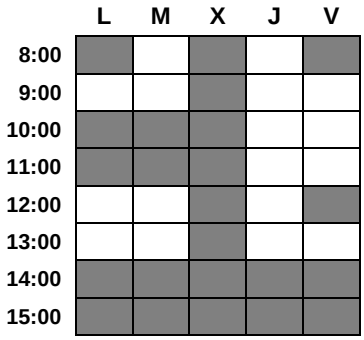
Si no hay posibilidad de encontrar un hueco para la reunión, el programa escribirá en pantalla `'No se puede celebrar la reunión'` y en caso contrario escribirá por pantalla el día y la hora en que se ha programado (ver ejemplo) y **actualizará** el fichero **`'agendas.dat'`** para que la hora del día de la reunión pase a estar marcada como ocupada para todos los directivos.

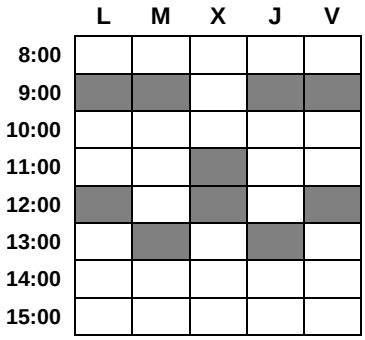
Se permite al alumno elegir una de entre las dos siguientes definiciones de tipos de datos:

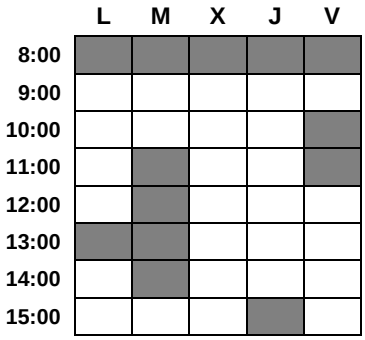
```
const
  HorIni = ...; { Hora inicio jornada }
  HorFin = ...; { Hora final jornada }
type
  TDiaSem = 1..5; { 1 -> Lunes, etc. }
  THora   = HorIni..HorFin;
  TAgenda = array[TDiaSem,THora] of boolean;
  TFichAgendas = file of TAgenda;
```

```
const
  HorIni = ...; { Hora inicio jornada }
  HorFin = ...; { Hora final jornada }
type
  TDiaSem = 1..5; { 1 -> Lunes, etc. }
  THora   = HorIni..HorFin;
  TDisponible = set of THora
  TAgenda = array[TDiaSem] of TDisponible;
  TFichAgendas = file of TAgenda;
```

Ejemplo: Si **HorIni** = 8 y **HorFin** = 15, y el fichero contiene las siguientes agendas de tres directivos (un cuadrado en blanco significa hora disponible), el programa escribiría el mensaje **`'La reunión será el jueves a las 10:00'`** y actualizará las 3 agendas en consecuencia.







Nota: El número de directivos está en el rango $[0..\infty)$

2. (1 *punto*) El **coeficiente binomial** $\binom{n}{k}$ representa el número de formas distintas de elegir *k* elementos de un conjunto de *n* elementos. Los valores de *n* y *k* son enteros nulos o positivos y el valor de *k* es menor o igual que el de *n*. Usar la siguiente definición recursiva

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \quad \binom{n}{0} = \binom{n}{n} = 1$$

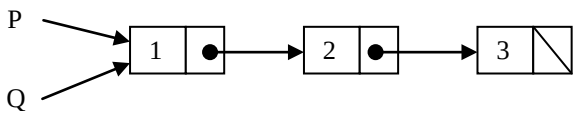
para crear una función **recursiva** que calcule el coeficiente binomial de *n* y *k*.

3. (2 *puntos*) Crear una función que reciba dos cadenas de caracteres (de tipo **string**) y devuelva un valor lógico indicando si las cadenas son **anagramas**. Una cadena es un anagrama de otra si contiene las mismas **letras** (y el mismo número de cada una de ellas) que la otra aunque ordenadas de forma distinta. Cualquier carácter que no sea letra no se tiene en cuenta, y se entiende como **letra** cualquier carácter en el rango 'A'..'Z' (alfabeto anglosajón). **Ejemplos**:

- `'TOM SORVOLO RYDDLE'` es un anagrama de `'SOY LORD VOLDEMORT'`
- `'HOFFS-DRAWLAR'` es un anagrama de `'FLASHFORWARD'`
- `'PASCAL'` **no** es un anagrama de `'SACA LA P'` (el carácter 'A' no aparece el mismo número de veces)

4. (1 *punto*) Dadas las siguientes variables, y suponiendo que en un momento dado de la ejecución se tiene el esquema de la derecha:

```
type
  PNodo = ^TNodo;
  TNodo = record
            Dato: integer;
            Sig:  PNodo;
          end;
var
  P,Q,R : PNodo;
```



Escribir las sentencias necesarias para pasar al esquema siguiente (**Nota**: No se puede cambiar el campo Dato):