Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

7 de febrero de 2019
Examen de **Fundamentos de Informática**
Grado en Tecnologías Industriales
Duración total del examen: **3 horas**

NOTA. Responde a cada ejercicio en una hoja diferente.

Ejercicio 1 [3 puntos]

Un número *primo equilibrado* es un número primo que cumple que es igual a la media aritmética de sus primos predecesor y sucesor. Dicho de otra forma, un número primo p_n es equilibrado si cumple:

$$p_n = \frac{p_{n-1} + p_{n+1}}{2}\ , \tag{1}$$

donde n es el índice (o número de orden) en el conjunto ordenado de los primos naturales.

Por ejemplo, el 5 (el tercer primo) es un primo equilibrado porque es igual a la media aritmética del 3 (el segundo primo, y por tanto el primo predecesor de 5) y el 7 (el cuarto primo, y por tanto el primo sucesor de 5): $(3+7)/2 = 5$. Los primeros primos equilibrados son: 5, 53, 157, 173, 211, 257, 263.

Se pide desarrollar un programa en Pascal que, dado un intervalo $[a, b]$, con a y b dos números naturales introducidos por teclado tales que $a \geq 0$ y $b \geq a$, muestre por pantalla todos los primos equilibrados pertenecientes a dicho intervalo, separados por espacios en blanco.

Observaciones:

- Se puede utilizar el operador de división / con operandos de tipo entero, en cuyo caso el resultado será de tipo real.
- Se pueden utilizar los operadores de relación = y <> con un operando de tipo real y otro de tipo entero.
- El 1 no es un número primo.

Ejercicio 2 [3.5 puntos]

Un estudiante está intentando organizarse su horario semanal para ver en qué asignaturas puede matricularse durante semestre concreto, incluyendo los 5 días laborables de la semana (de lunes a viernes). Para ello escribe en un fichero de texto `horarios.txt` todas las actividades planeadas. El fichero de texto contiene, por cada línea y separados por espacios:

- El día de la semana con un carácter: la inicial del día de la semana, excepto el miércoles (X).
- Horas de entrada y salida: dos números entreros (toda actividad comienza y acaba a en punto).
- Una descripción de la actividad a realizar: hasta 255 caracteres de texto, que llega hasta fin de línea, representable como un `String` de Pascal.

Cada actividad parece en una línea independiente en dicho fichero, y las actividades no están dispuestas en ningún orden concreto ni agrupadas de ninguna manera específica. La resolución temporal de las actividades es de una hora, y comienzan y acaban a horas en punto. Esto implica que no hay actividades que duren una fracción de hora y que un día dispone de exactamente 24 huecos en los que disponer actividades.

Se pide desarrollar un programa Pascal que lea el fichero `horarios.txt` y muestre por pantalla todos los conflictos entre actividades, sin ningún orden específico, incluyendo el día y la hora del conflicto, así como las descripciones de las dos actividades que entran en conflicto. Se puede asumir que el fichero de texto sigue exactamente la estructura predicha: el dia siempre será un carácter correcto y no habrá

horas que estén fuera del rango del horario normal de un día. Si hay un conflicto entre más de dos actividades el mismo día y hora, no es necesario mostrar todas las actividades en conflicto (solo dos de ellas).

Ejemplo de ejecución:

horarios.txt	**Salida del programa**
X 18 20 Fundamentos de Informatica L 14 15 Problemas Fundamentos de Informatica J 14 15 Fundamentos de Informatica L 14 16 Practicas de Matematicas I M 10 13 Practicas de Estadistica M 11 13 Seminario V 16 20 Academia V 15 17 Matematicas I	Conflicto los lunes de 14h a 15h : Practicas de Matematicas I Problemas Fundamentos de Informatica Conflicto los martes de 11h a 12h : Seminario Practicas de Estadistica Conflicto los martes de 12h a 13h : Seminario Practicas de Estadistica Conflicto los viernes de 16h a 17h : Matematicas I Academia

Ejercicio 3 [3.5 puntos]

En el Patinaje Artístico los patinadores se organizan por niveles, que para simplificar numeraremos de 1 a 5, siendo el 1 el más básico y el 5 el más avanzado. Para ascender de nivel hay que superar un Test (examen) en el que tres jueces votan individualmente SÍ o NO al ascenso, y el deportista sube de nivel si al menos dos le dan su voto favorable. Los del nivel más avanzado no se examinan más. Se desea implementar en un programa Pascal la gestión de los Tests.

Define los siguientes tipos de datos:

- `tpPatinador`: un deportista, del que nos interesa conocer su nombre (de no más de 255 caracteres de texto), su nivel, y los votos de los tres jueces en el Test.
- `tpGrupo`: una secuencia de patinadores de longitud variable, con un máximo de 1000.
- `tpNiveles`: una tabla que contiene un grupo de patinadores para cada nivel del 1 al 5.

Se pide escribir los procedimientos o funciones necesarios para:

- Dada una secuencia (de tipo `tpGrupo`) con TODOS los patinadores que participan en el Test mezclados, generar la tabla de `tpNiveles` con los deportistas separados en grupos por niveles:

```
procedure SeparaNiveles(??? todos: tpGrupo; ??? niveles: tpNiveles);
```

- Saber si un patinador ha superado el Test, a partir de los votos de los jueces:

```
function Aprueba(??? p: tpPatinador): ???;
```

- Generar la nueva tabla de patinadores (de tipo `tpNiveles`) despues del Test, a partir de la `original` de antes del Test, para subir de nivel a los que han aprobado:

```
procedure ActualizaNiveles(??? original: tpNiveles;
                           ??? nueva:    tpNiveles);
```

  La `original` ya contiene los tres votos de los jueces para cada patinador, pero sin actualizar el nivel; la nueva debe tener los patinadores con su nivel actualizado tras el Test, los votos anulados, y estar en su grupo correspondiente. Cada nivel de la tabla nueva puede rellenarse con los que no han aprobado ese nivel de la `original`, más los que han aprobado del nivel anterior. Hay que tener en cuenta que no hay un nivel inferior al 1 y que los de nivel 5 no se examinan.

Además de los que se piden explícitamente, se pueden implementar los procedimientos y funciones que se considere adecuado.