UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación
E.T.S.I. Informática

Fundamentos de la Programación
Examen 1ª Convocatoria Ordinaria **02/02/18**

Apellidos, Nombre:

Titulación:

Grupo:

Código PC usado:

NOTAS PARA LA REALIZACIÓN DEL EXAMEN:

- La solución se almacenará en la carpeta **EXAMENFEBFP**, dentro de **Documentos**. Si la carpeta ya existe, debe borrarse todo su contenido. En otro caso, debe crearse.
- Los nombres de los ficheros con la solución para los ejercicios 1, 2, 3 y 4 serán **ejercicio1.cpp**, **ejercicio2.cpp, ejercicio3.cpp** y **ejercicio4.cpp,** respectivamente.
- Al inicio del contenido de cada fichero deberá aparecer un comentario con **el nombre del alumno, titulación, grupo y código del equipo** que se está utilizando (cada dato en una línea diferente)**.**
- Una vez terminado el examen, se subirán los ficheros ***.cpp** a la tarea creada en el **campus virtual** para ello.
- **No está permitido**:
  - Utilizar documentación electrónica o impresa.
  - Intercambiar documentación con otros compañeros.
  - Utilizar soportes de almacenamiento.
  - Utilizar dispositivos electrónicos (móviles, tablets, …)

(**1 pto) 1.- Diseña** una **función** *valorDominante* que recibe como parámetro un array completo de tamaño TAM (una constante establecida) de valores naturales y devuelve el valor dominante del mismo o -1 en caso de que dicho valor no exista. El valor dominante de un array de naturales es el elemento que se repite un número de veces mayor que la mitad del tamaño del array.

Importante: sólo se completará el código de la función *valorDominante* en el fichero *ejercicio1.cpp* proporcionado en el campus virtual. Puedes añadir más procedimientos o funciones si lo estimas necesario. No se debe modificar el resto del código proporcionado. La puntuación de este problema será de 1 punto sólo en el caso de que la búsqueda funcione correctamente y se haga de forma que si se encuentra el valor dominante, se detenga la misma. En otro caso la puntuación será de 0 puntos.

La ejecución del código suministrado (tras diseñar la función solicitada) será:

```
El elemento dominante del primer array es: 3
El elemento dominante del segundo array es: -1
El elemento dominante del tercer array es: 4
```

(**2 ptos) 2.- Diseña** un **algoritmo** que lea de teclado números naturales para completar una matriz de tamaño F x C (siendo F y C dos constantes naturales establecidas) y muestre por pantalla todas las cimas que existen en la matriz. Para cada cima se mostrarán la fila, la columna y valor de la misma.

Una cima es un elemento de la matriz que es mayor o igual que los elementos vecinos que están a su izquierda, derecha, arriba y abajo (no se tienen en cuenta las diagonales). Un elemento puede tener 2, 3 o 4 vecinos según sea esquina, borde-no-esquina o no-borde-no-esquina, respectivamente.

Ejemplos (F = 3 y C = 3):

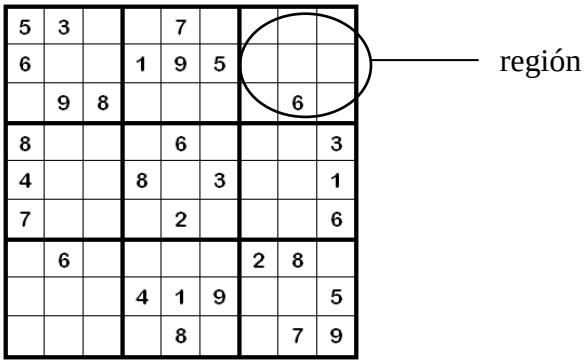
Entrada: 4 5 3 6 2 2 1 8 7 Salida: Las cimas de la matriz son: Fila 0 columna 1 valor 5 Fila 1 columna 0 valor 6 Fila 2 columna 1 valor 8	Entrada: 4 4 4 4 4 4 4 4 4 Salida: Las cimas de la matriz son: Fila 0 columna 0 valor 4 Fila 0 columna 1 valor 4 Fila 0 columna 2 valor 4 Fila 1 columna 0 valor 4 Fila 1 columna 1 valor 4 Fila 1 columna 2 valor 4 Fila 2 columna 0 valor 4 Fila 2 columna 1 valor 4 Fila 2 columna 2 valor 4
Entrada: 1 2 3 4 5 6 7 8 9 Salida: Las cimas de la matriz son: Fila 2 columna 2 valor 9	Entrada: 9 8 7 6 5 4 3 2 1 Salida: Las cimas de la matriz son: Fila 0 columna 0 valor 9

(**3.5 ptos) 3.- Diseña** un **algoritmo** que lea de teclado números enteros para completar una matriz de tamaño 9 x 9. Posteriormente comprobará si dicha matriz constituye un tablero de sudoku válido o no y mostrará por pantalla el mensaje correspondiente.

Un tablero de sudoku es una matriz de 9 x 9 casillas dividida en 9 regiones de 3 x 3 casillas cada una (ver ejemplo de abajo). Las reglas para que un tablero de sudoku sea válido son las siguientes:

- Cada *casilla* almacena un número comprendido entre 1 y 9 (ambos inclusive) o bien está vacía (para nosotros estará vacía si contiene un 0)
- En una misma *fila* no puede haber números repetidos (los 0 no cuentan).
- En una misma *columna* no puede haber números repetidos (los 0 no cuentan).
- En una misma *región* no puede haber números repetidos (los 0 no cuentan).

Un ejemplo de tablero de sudoku válido sería el siguiente:



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(**3.5 ptos) 4.- Diseña** un **algoritmo** que lea de teclado un patrón y un texto, y muestre por pantalla un listado de todas las palabras del texto indicando el número de caracteres que tienen en común con el patrón. Habrá que asegurarse que el patrón tenga una longitud determinada (definida por una constante (en el ejemplo de abajo tiene el valor 5)) y que no contenga letras repetidas. El orden en el que aparezcan las letras en el patrón y en la palabra no es importante. Si una letra del patrón aparece varias veces en una palabra del texto solo se cuenta una vez. **En la salida no habrá palabras repetidas**.

Ejemplo:

Entrada:

```
Introduzca un patron (long = 5, sin letras repetidas): EL
Introduzca un patron (long = 5, sin letras repetidas): CANTA
Introduzca un patron (long = 5, sin letras repetidas): CANTO

Introduzca un texto (FIN para terminar): ANTERIORMENTE IBA A TRABAJAR
EN TREN PERO AHORA VOY A TRABAJAR EN AUTOMOVIL FIN
```

Salida:

```
Palabras y numero de letras que coinciden con el patron:
ANTERIORMENTE 4
IBA 1
A 1
TRABAJAR 2
EN 1
TREN 2
PERO 1
AHORA 2
VOY 1
AUTOMOVIL 3
```

NOTAS:

- El texto contiene un número indefinido de palabras.
- El texto termina con la palabra FIN.
- Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos mayúsculas).
- El patrón tendrá una longitud de TAM_CAR (una constante) caracteres.
- En el texto habrá un número máximo MAX_PAL_DIST (una constante) de palabras distintas.
- El carácter separador de palabras es el espacio en blanco.