Fundamentos de Programación
Relación de ejercicios 5

1. Definir una función recursiva para sumar los dígitos de un entero positivo.

2. Definid una función recursiva con prototipo

   ```
   int MCD(int un_entero, int otro_entero)
   ```

   para que calcule el máximo común divisor entre dos enteros, aplicando el algoritmo de Euclides, que nos dice lo siguiente:

   ```
   MCD(a, 0) = a
   MCD(a, b) = MCD(b, a % b)
   ```

   dónde $a\%b$ es el resto de la división entera entre a y b. Implementad el algoritmo suponiendo que a es mayor que b. A continuación, comprobad que también funciona en el caso contrario. ¿Por qué?

3. Definir una función recursiva para construir un vector de 0 y 1 con la representación en binario de un entero. En la página web

   `http://es.wikipedia.org/wiki/Sistema_binario`

   se puede consultar cómo pasar un número en decimal a binario (el mismo procedimiento para pasar un decimal a binario se aplica sobre cualquier otra base menor o igual que 10). El bit más significativo (el de la mayor potencia de 2) debe ir en la posición 0 del vector, mientras que el menos significativo debe ir en la última posición del vector.

4. Definir una función recursiva a la que se le pasen dos valores enteros n y k y devuelva como resultado el valor k-ésimo comenzando por la derecha del número n. Por ejemplo para $n = 427$ y $k = 3$ el resultado sería 4. Si k es mayor que el número de dígitos de n la función devolverá el valor 0. Por ejemplo, para $n = 23$ y $k = 5$ el resultado es 0.

5. Cread sendas funciones recursivas para:

   a) Calcular la división entera entre dos números enteros positivos.

   b) Calcular el resto de la división entera de dos números enteros positivos.

   Pasad como parámetros a cada función los dos enteros, y suponed que en la llamada siempre pasarán números positivos (es la precondición de la función). En la definición de las funciones no pueden utilizarse los operadores de división, multiplicación ni módulo.

6. Cread una función recursiva con la siguiente cabecera

   ```
   bool ContieneDesdeInicio(string cadena_grande, string cadena_pequena);
   ```

   que compruebe si la cadena `cadena_pequena` se encuentra al inicio de la cadena `cadena_grande` (desde la posición cero)

7. Cread una función recursiva con la siguiente cabecera

   ```
   bool Contiene(string cadena_grande, string cadena_pequena)
   ```

   que compruebe si la cadena `cadena_pequena` se encuentra dentro de la cadena `cadena_grande`. En la definición de esta función, se puede llamar a la función `ContieneDesdeInicio` del ejercicio 8.

8. Construir una función recursiva que muestre en la salida estándar los m mayores números pares que sean menores o iguales que un número n. El prototipo sería el siguiente:

   ```
   void MayoresPares(int m, int n);
   ```

   Por ejemplo:

   `MayoresPares(3,6)`, imprimirá en pantalla 6, 4, 2.

   `MayoresPares(4,3)`, imprimirá en pantalla 2, 0, -2, -4.

   `MayoresPares(0,8)`, no imprimirá nada en pantalla.

9. Cree una función recursiva que, para dos valores enteros positivos n y k, calcule cuántos divisores propios positivos tiene n que sean estrictamente menores que k. Un número d es divisor propio de m si d divide a m y d es distinto de m (el 1 es divisor propio de cualquier número distinto de 1). La cabecera de la función ha de ser la siguiente:

   ```
   int divisoresMenores(int n, int k);
   ```

   Por ejemplo, `divisoresMenores(10, 6)` debe devolver 3 ya que 1, 2 y 5 son los dos divisores propios positivos de 10 estrictamente menores que 6. Observe que, para cualquier número p mayor que 1, el resultado devuelto por `divisoresMenores(p,p)` debe ser 1 si p es primo.

10. Implemente una función recursiva que acepte un entero y devuelva el vector de enteros menor o iguales que él (hasta llegar al 0). Si se introduce un valor negativo tendrá que devolver un vector formado por dicho negativo. Por ejemplo, si el entero es 4, la función debe devolver el vector $\{0, 1, 2, 3, 4\}$. Si el entero es -4, la función debe devolver el vector $\{-4\}$. No puede usarse ninguna estructura repetitiva (bucles while o for) y se recomienda usar un `vector` de la STL.

11. Definir una función recursiva que acepte un `long long` y devuelva un vector de caracteres con los dígitos del mismo incluyendo los puntos de separación cada tres cifras decimales. Por ejemplo, para el entero 23456789, la función devolvería el vector de caracteres 23.456.789.

12. El método de bisección usado para calcular de forma aproximada un punto de corte de una función $f : \mathbb{R} \to \mathbb{R}$ con el eje de abscisas (una raíz de la función) se puede describir como sigue:

    Sea $f(x)$ una función real continua en $[i, d]$, donde $f(i)$ y $f(d)$ tienen distinto signo y sea ϵ una constante real pequeña (del orden de 10^5, por ejemplo). Entonces, seguir el siguiente procedimiento.

    1. Calcular m, el punto medio entre i y d,
    2. Si $|i - d| < \epsilon$, terminar.
    3. Si $f(m)$ tiene igual signo que $f(i)$, repetir considerando el intervalo $[m, d]$.
    4. Si $f(m)$ tiene igual signo que $f(d)$, repetir considerando el intervalo $[i, m]$.

    Implemente una función recursiva que reciba como datos de entrada los extremos i y d del intervalo, así como el valor de ϵ y devuelva una raíz de la función. Se supone que la función $f(x)$ ya está definida con anterioridad.

    Nota: recordemos que esto es consecuencia del teorema de Bolzano que, en las condiciones anteriores, asegura la existencia de una raíz en el intervalo $[i, d]$. Sin embargo no puede asegurar que sea única. Para ello se suele completar con el teorema de Rolle:

    Teorema de Rolle. Si una función continua y derivable f definida en un intervalo $[a, b]$ verifica que $f(a) = f(b)$ entonces existe un valor c en dicho intervalo ($c \in (a, b)$) tal que $f'(c) = 0$.

De este modo, si la ecuación $f(x) = 0$ tiene dos soluciones x_1 y x_2 entonces según el teorema de Rolle la derivada de f tiene también una raíz en el intervalo. Así, si se sabe que la derivada no tiene ceros en un intervalo no habrá más de una solución de la ecuación $f(x) = 0$ en dicho intervalo. Dicho de otro modo, si la función es estrictamente monótona.

13. Realizar una función recursiva para multiplicar dos enteros según el método de multiplicación rusa (ver la relación 2).

14. Implementar una función recursiva que devuelva la posición de 8 reinas en un tablero de ajedrez sin que se hagan jaque entre sí. Dos reinas se hacen jaque si se encuentran en la misma fila, en la misma columna o en la misma diagonal (tanto derecha como izquierda).

15. Implementar una función recursiva que devuelva el orden de las casillas que debe pisar un caballo de ajedrez, comenzando desde cualquier casilla inicial dada, para que pueda recorrer todo el tablero de ajedrez sin pisar dos veces la misma casilla.

16. Implementar una clase `Laberinto` que represente un laberinto, formado por una matriz de enteros, donde cada componente de la matriz posee un valor si tiene un muro (1, por ejemplo) y otro valor si está vacía (0, por ejemplo) y se puede pasar por ella. El laberinto deberá tener una entrada y una salida. A continuación, implementa una función miembro recursiva que, dada una casilla inicial (la entrada al laberinto, marcada con 2, por ejemplo), muestre por pantalla el camino a seguir desde dicha casilla hasta la salida del laberinto (marcada con un 4, por ejemplo), en caso de existir.

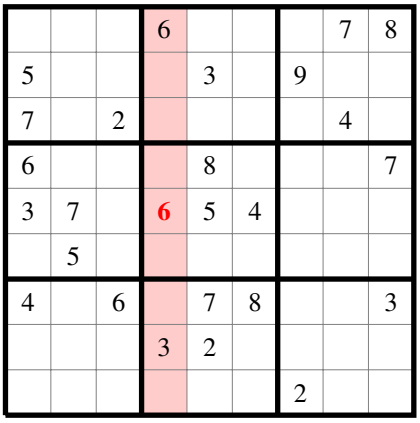
```
1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 1 1 0 1 1 1 1 0 0 1
1 0 0 1 1 1 0 0 1 0 0 1
1 0 0 0 0 0 0 0 1 0 0 4
1 1 2 1 1 1 1 1 1 1 1 1
```

17. Implementar una clase `Sudoku` que represente un sudoku. Un sudoku, en su formato original, consiste en una cuadrícula 9×9, que está parcialmente rellena con números enteros del 1 al 9, y que se ha de completar colocando dichos enteros siguiendo las siguientes reglas:

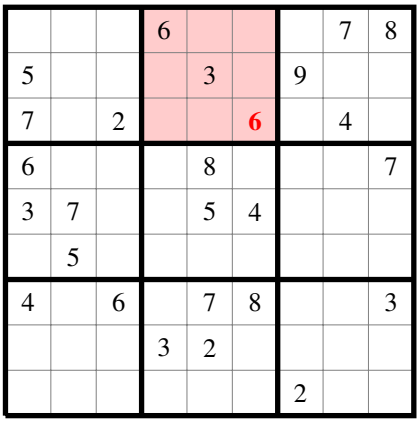
    - Un número NO puede aparecer dos veces en la misma fila

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

- Un número NO puede aparecer dos veces en la misma columna



- Un número NO puede aparecer dos veces en la misma región (subcuadrícula 3×3)



Implementar una función miembro recursiva que devuelva una solución del sudoku, si es que existe.