# Práctica 3

**Importante:** Los ejercicios deben entregarse a través de web (**Domjudge y Blackboard**). Cada ejercicio deberá ir en un fichero con nombre:

*<<nombreEjercicio>>.cpp*

donde <<nombreEjercicio>> es el nombre indicado en negrita antes de cada ejercicio.

**La fecha de entrega:** consultar la página de la actividad en blackboard

**Combinaciones (1 Punto):** Escribe un programa para calcular las posibles combinaciones de n elementos tomados en grupos de r usando una solución recursiva:

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

El **input** se leerá con el formato `"%d %d"`. El primer entero estará referido al valor de **n** y el segundo al de **r**.
El **output** utilizará el formato `"%d\n"` que corresponderá al número de combinaciones posibles. Si las entradas no cumplen las precondiciones deberá imprimir "ERROR" y continuar leyendo. Si se recibe un valor n negativo se deberá terminar el programa.

| Input: | Output: |
|---|---|
| 6 4<br>7 3<br>-1 0 | 15<br>35 |

**Palíndromo (1 Punto):** Escribir un programa recursivo que lea una cadena de caracteres e indique si es un palíndromo, es decir, si la palabra se puede leer igual de izquierda a derecha que de derecha a izquierda. El output será un 1 si es un palíndromo y un 0 en caso contrario.

| Input: | Output: |
|---|---|
| Anilina | `1` |

La máxima longitud de la cadena de caracteres es 20.

**RaízCuadrada (2 Puntos):** Escribe un programa que permita calcular la raíz cuadrada de un número de forma recursiva mediante aproximaciones sucesivas. Por ejemplo, para calcular la raíz cuadrada del número 16 con un error menor que 0.1. Inicialmente, se tomará el valor medio entre 0 y el número objetivo (16), es decir 8. Si el error entre el

cuadrado de este número (8x8=64) y el número objetivo (16) es menor que el error indicado se deberá terminar. En este caso el error es mayor (64-16>0.1) así que será necesario continuar y para ello se cambiará el rango de búsqueda dependiendo de si el cuadrado era mayor o menor que el objetivo. En este caso el valor de nuestra primera aproximación (8) al cuadrado es 64 que es bastante superior que 16 así que la nueva aproximación será el valor medio entre 0 y 8 que es 4. Al calcular el cuadrado de 4 nos da 16 que coincide que el número objetivo, por tanto el error será menor que 0.1 (16-16<0.1) así que el código podrá terminar.

El **input** se leerá con el formato `"%f %f"`. El primer valor se refiere al número del que se quiere calcular la aproximación de raíz cuadrada y el segundo al error con el que se tiene que calcular dicha aproximación.
El **output** utilizará el formato `"%f\n"` por cada aproximación calculada hasta alcanzar una aproximación que sea menor que el error indicado. Si las entradas no cumplen las precondiciones deberá imprimir "ERROR".

| Input: | Output: |
|---|---|
| 2 0.01 | 1<br>1.5<br>1.25<br>1.375<br>1.4375<br>1.40625<br>1.42188<br>1.41406 |

**Determinante (4 Puntos):** Añadir al código desarrollado en la Práctica 1, el siguiente método:

**-double calcularDeterminante()**.Este método permitirá calcular el determinante de la matriz mediante el método de los adjuntos realizando llamadas recursivas con matrices de dimensiones inferiores como se puede observar en la siguiente figura:

$$\begin{vmatrix} 3 & 2 & 1 \\ 4 & -3 & 2 \\ 1 & -1 & 2 \end{vmatrix} = 3 \cdot \begin{vmatrix} -3 & 2 \\ -1 & 2 \end{vmatrix} - 2 \cdot \begin{vmatrix} 4 & 2 \\ 1 & 2 \end{vmatrix} + 1 \cdot \begin{vmatrix} 4 & -3 \\ 1 & -1 \end{vmatrix} =$$

$$= 3 \cdot (-4) - 2 \cdot 6 + 1 \cdot (-1) = -25$$

NOTA: Para cada una de las funciones implementados se deberá incluir una pequeña descripción de su funcionamiento, sus precondiciones mediante assertdomjudge si las hubiera y el análisis de su complejidad temporal y espacial. Esta información deberá incluirse en la cabecera de cada función. Para los ejercicios **Raíz Cuadrada** y **Determinante** no será necesario calcular las complejidades temporales y espaciales pero serán valorables las aproximaciones.