Objetivo de la práctica:
Practicar el paso de parámetros por referencia y las funciones recursivas.
Nota: Todos los ejercicios son obligatorios excepto el último (Torres de Hanoi).

Conceptos sobre funciones con parámetros por referencia

- Las funciones, por norma general, resuelven algún subproblema requerido desde el programa principal o desde otra función. Para ello, en la mayoría de ocasiones, necesitarán **recibir parámetros** (a modo de variables) desde el programa que las invoca, de forma que puedan operar con ellos y conseguir su objetivo. **El paso de estos parámetros/variables** puede implementarse en C++ de dos formas distintas:

  - En primer lugar, todo sub/programa que llama a una función puede enviarle los parámetros necesarios por **valor**. En este caso, la función al recibir los parámetros se hará una **copia** de los mismos y operará con esta copia, de modo que, las variables del sub/programa principal enviadas como parámetros no se verán afectadas (aunque se modifiquen en la función, realmente se modifica la copia). Por esta razón se dice que los parámetros pasados por valor son, *parámetros de entrada* a la función.

  - En segundo lugar, en C++ se pueden pasar parámetros/variables por **referencia**. En este caso, se utilizan las mismas variables (direcciones físicas de memoria). Al no hacer copia de las variables de entrada, debemos tener en cuenta que cualquier cambio, en el valor de alguna variable, se mantendrá una vez finalizada la función. Esta modalidad de paso de parámetros suele utilizarse para implementar funciones que devuelvan más de un valor, ya que ahora podemos definir funciones que devuelvan: a) el *valor de retorno* por defecto, y b) las variables por referencia o *parámetros de salida* empleados.

  - Para diferenciar ambos tipos de parámetros (ya que ambos pueden aparecer indistintamente en la cabecera de la función), los **parámetros por referencia utilizan el símbolo &**, tanto en el prototipo como en la cabecera de la función.

```
Ej: <tipo> funcion (<tipo_p> &param); // Parámetro param por referencia
Ej: <tipo> funcion (<tipo_p> param);  // Parámetro param por valor
```

PROBLEMAS (pasos por referencia)

1. Escribe una función que reciba dos caracteres y los intercambie (utiliza 2 parámetros por referencia). Imprime los caracteres antes y después de llamar a la función.

2. (*)Escribe una función que genere y devuelva combinaciones de la lotería primitiva (6 números y el complementario). Ayuda: realizar una función que genere un número aleatorio en un intervalo determinado utilizando la función **int rand()** (devuelve un número entero en el intervalo [**0,RAND_MAX**]) .

3. (*)Escribe una calculadora sencilla para vectores 2D situados en el origen de coordenadas. El programa pedirá al usuario la operación a realizar (+, -, *, /), así como los operandos necesarios (puntos x1, y1, x2, y2), y devolverá el resultado de la operación solicitada.

4. (*)Dadas dos rectas representadas por 2 pares de vértices escribir una función que:
   - Si son paralelas devuelva **false**. Para saber si son paralelas calcular los vectores direccionales unitarios de cada una de las rectas y comprobar si son iguales
   - Si no son paralelas devolver **cierto** y el punto de intersección de las mismas basándonos en el despeje de la x e y de la ecuación que describe una recta.

Conceptos sobre funciones recursivas

Dentro del código de una función recursiva hay una sentencia o expresión donde aparecerá la llamada a la misma función. Para no generar infinitas llamadas (dado que la función se llama a sí misma) necesitamos tener en cuenta lo siguiente:

- Una función recursiva resolverá un problema de talla N, considerando resuelto el problema de talla N – 1. Por tanto, **los valores de los parámetros de la función se modifican** de una llamada a otra.

  ej: `Factorial(N) = N * Factorial (N-1);  // problema_de_talla_N = N * problema_de_talla_N-1`

- Antes de que se produzca la recursión, debe aparecer la **condición de parada** (caso base), que estará relacionada con alguno de los parámetros de la función (ej: `factorial(1) = 1`). Al detectar este caso, no se producirán más llamadas recursivas y se devolverá el valor correspondiente.

PROBLEMAS (funciones recursivas)

4. (*)Escribe una función recursiva para sumar 2 enteros. Dentro de la función sólo se podrán utilizar incrementos unitarios (++) y decrementos unitarios (--) .

5. Escribe otra función recursiva capaz de multiplicar 2 números enteros (en este caso sólo se podrá utilizar el operador suma, y no el de multiplicación).

6. La función exponencial, x^y siendo *x, y* números enteros, puede implementarse de forma recursiva, dado que
   $x^y = x * x^{y-1}$. Implementa una función recursiva que calcule la función exponencial.

   6a .- Mediante la estrategia conocida como *divide y vencerás* nos podemos ahorrar un número de multiplicaciones considerable, para ello:

   - elige un entero $k<y$ (por ejemplo la mitad : k = y/2)
   - $x^y = x^k * x^{y-k}$ (mismo problema pero con talla más pequeña) ; si k = y/2 es PAR tenemos que $x^y = x^k * x^k$, si k es IMPAR $x^y = x^k * x^{k+1}$, con lo que bastará con calcular una sola vez x^k y multiplicar el valor de x^k por x para obtener la potencia deseada (k+1).

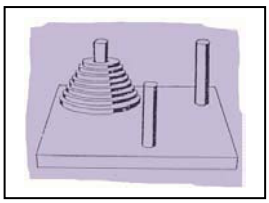
   Implementa otra función recursiva que calcule la exponencial con este método y añade un contador de llamadas recursivas para comprobar el número de multiplicaciones ahorradas por este método.

LA LEYENDA DE LAS TORRES DE HANOI *(Opcional)*

Dice la leyenda que, al crear el mundo, Dios situó sobre la Tierra tres varillas de diamante y sesenta y cuatro discos de oro. Los discos son todos de diferente tamaño e inicialmente fueron colocados en orden decreciente de diámetros sobre la primera de las varillas. También creó Dios un monasterio cuyos monjes tienen la tarea de trasladar todos los discos desde la primera varilla a la tercera. La única operación permitida es mover un disco de una varilla a otra cualquiera, pero con la condición de que no se puede situar encima de un disco otro de diámetro mayor. La leyenda dice también que

cuando los monjes terminen su tarea, el mundo se acabará. Implementa la función recursiva que calcule los movimientos necesarios para mover N discos de acuerdo con las reglas anteriores y rellena la tabla siguiente:

```
void Hanoi(int N, int desde, int hacia, int usando)
```



Número de Discos:	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Número de Movimientos:	**1**	**3**					

- Suponiendo que los monjes son capaces de realizar un movimiento por segundo, ¿sabrías calcular en cuánto tiempo acabará el mundo, según la leyenda?.