Hoja de ejercicios del Tema 4

1. Dado el siguiente código de C++ (suponiendo incluida la biblioteca iostream):

```
int main() {
   int a = 1, b = 2, c = 3;
   otro(a, b, c);
   cout << "main dice: ";
   cout << "a = " << a << " b = " << b << " c = " << c << endl;
   otro(b, a ,c);
   cout << "main dice: ";
   cout << "a = " << a << " b = " << b << " c = " << c << endl;
   return 0;
}

void otro(int a, int &b, int c) {
   c= a + b;
   b = a;
   a = c;
   cout << "otro dice: ";
   cout << "a = " << a << " b = " << b << " c = " << c << endl;
}
```

Indica, sin usar un compilador, cuál será la salida del programa.

2. Dado el siguiente programa en C++:

```
int main() {
   char a = 'B', b = 'C', c = 'A', d = 'D';
   vueltas(a, b, c, d);   // 1ª llamada
   vueltas('A', b, c, a); // 2ª llamada
   vueltas('D', c, c, a); // 3ª llamada
   return 0;
}

void vueltas (char a, char &b, char &d, char &c) {
   b = a;
   c = d;
   d = 'B';
}
```

Rellena la siguiente tabla con el valor de las variables del programa principal después de cada llamada al procedimiento `vueltas()`.

	a	b	c	d
Valor inicial	'B'	'C'	'A'	'D'
Después de la 1ª llamada				
Después de la 2ª llamada				
Después de la 3ª llamada				

3. Escribe un procedimiento en C++ que tenga como entrada un número entero positivo y que escriba en la pantalla una tabla como la siguiente, en la que se ha supuesto que el argumento utilizado para llamar al procedimiento es 4.

```
1     2     3     4      10
2     4     6     8      20
3     6     9    12      30
4     8    12    16      40
```

En la última columna se muestra el resultado de sumar todos los elementos anteriores de la fila correspondiente.

4. Escribe un procedimiento en C++ que encuentre y muestre todos los números de tres cifras en los que la suma de los cubos de sus dígitos sea igual al propio número. Ejemplo : $153 \equiv 1^3 + 5^3 + 3^3 = 1 + 125 + 27$

5. Escribe una función `lecturaConLimites()` que reciba como parámetros dos valores de tipo `int`. La función pedirá al usuario que introduzca un número entero tantas veces como sea necesario hasta que el número suministrado se encuentre dentro del intervalo determinado por los datos de entrada (*puede darle pistas*). La función devolverá el último número leído. Escribe un programa principal que use dicha función para pedir al usuario un número entre 10 y 20.

6. Escribe una función en C++ que acepte un valor de tipo `double` y devuelva el número redondeado a la centésima más cercana. Escribe un programa principal que pida al usuario un número real y lo muestre en la pantalla redondeado.

7. Escribe un programa en C++ que calcule cuántos números *perfectos* hay en un archivo de texto `datos.txt` con enteros (cada número en una línea y terminado en 0 como centinela). Se dice que un número (entero positivo) *n* es *perfecto*, si la suma de los divisores de *n* entre 1 y *n*-1 es igual a *n*. Por ejemplo, 6 es un número perfecto, pues sus divisores, incluyendo el 1 pero no el propio 6, son 1, 2 y 3, y sumados dan 6. En el ejercicio 21 del tema anterior vimos lo que son los números triangulares.

8. Matemáticamente se puede demostrar que todos los números perfectos son, además, triangulares. Escribe un programa en C++ que compruebe empíricamente que esto es cierto para todos los números perfectos menores de 100.000. En la hoja del tema anterior vimos lo que son los números triangulares.

9. Implementa una función `esPrimo(int n)` que devuelva `true` sí y sólo si `n` es primo. Utiliza la función implementada para:

a. Escribir todos los números primos que hay entre dos naturales dados.

b. Escribir el primer número primo mayor que un natural dado.

c. Contar los primos menores que un natural dado.

Crea un subprograma para cada opción y un menú para la aplicación.

10. Escribe una función `inverso()` que devuelva el resultado de invertir el entero positivo que reciba. Se entiende por invertir dar la vuelta a los dígitos del número (hallar su imagen especular); así, el inverso de 3952 es 2593.

Escribe una función `capicua()` que, haciendo uso de la función `inverso()`, devuelva un valor booleano que indique si el número entero positivo que recibe es o no capicúa.

Escribe un programa principal que solicite números enteros positivos e indique si son o no capicúas. El programa solicitará números hasta que se introduzca uno negativo.

11. Escribe un programa en C++ que permita manejar una lista de hasta 100 cantidades reales positivas. El programa empezará leyendo de un archivo `lista.txt` los valores que inicialmente va a contener (-1 como centinela final). A continuación permitirá al usuario realizar las siguientes acciones con la lista:

- ✓ Insertar una nueva cantidad al final de la lista
- ✓ Insertar una nueva cantidad al principio de la lista
- ✓ Eliminar una cantidad dada su posición en la lista
- ✓ Localizar una cantidad (indicando la posición en que se encuentra)
- ✓ Mostrar la lista de cantidades (una en cada línea precedida de su posición)
- ✓ Guardar en el archivo `lista.txt` la lista

Cada opción se implementará con un subprograma y habrá un menú de opciones. Habrá una función que indique si la lista está llena.

12. En combinatoria, el número de variaciones de x elementos de orden y, $V_{x,y}$, ($x > 0$, $0 < y \leq x$), el número de permutaciones de x elementos, P_x, ($x>0$) y el número de combinaciones de x elementos de orden y, $C_{x,y}$, ($x > 0$, $0 < y \leq x$) se obtienen mediante las siguientes fórmulas:

$$V_{x,y} = x \cdot (x-1) \cdot (x-2) \cdot \ldots \cdot (x-y+2) \cdot (x-y+1)$$

$$P_x = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-2) \cdot (x-1) \cdot x$$

$$C_{x,y} = V_{x,y} / P_y$$

Implementa una función `variaciones()` que reciba dos números enteros x e y y calcule y devuelva el entero que representa $V_{x,y}$. Otra función `permutaciones()` que reciba un número entero x y calcule y devuelva el entero que representa P_x. Y otra función `combinaciones()` que reciba dos números enteros x e y y calcule y devuelva (haciendo uso de las dos funciones anteriores) el entero que representa $C_{x,y}$.

Implementa un programa principal que solicite al usuario parejas de enteros positivos x e y mientras que no sean ambos 0 y que para cada pareja compruebe que cumplen las condiciones necesarias para poder calcular $V_{x,y}$, P_x y $C_{x,y}$ ($x > 0$, $0 < y \leq x$) y si es así los calculen, haciendo uso de los subprogramas anteriores, y los muestren en la pantalla.

***NOTA**: Las tres funciones necesitan que los números que reciben como entrada cumplan las condiciones necesarias para poder calcular los valores correspondientes. Usa la función* `assert()` *durante la depuración.*