Fundamentos de Programación

Catálogo de ejercicios y soluciones

Félix Gómez Mármol
Mariano Albaladejo González

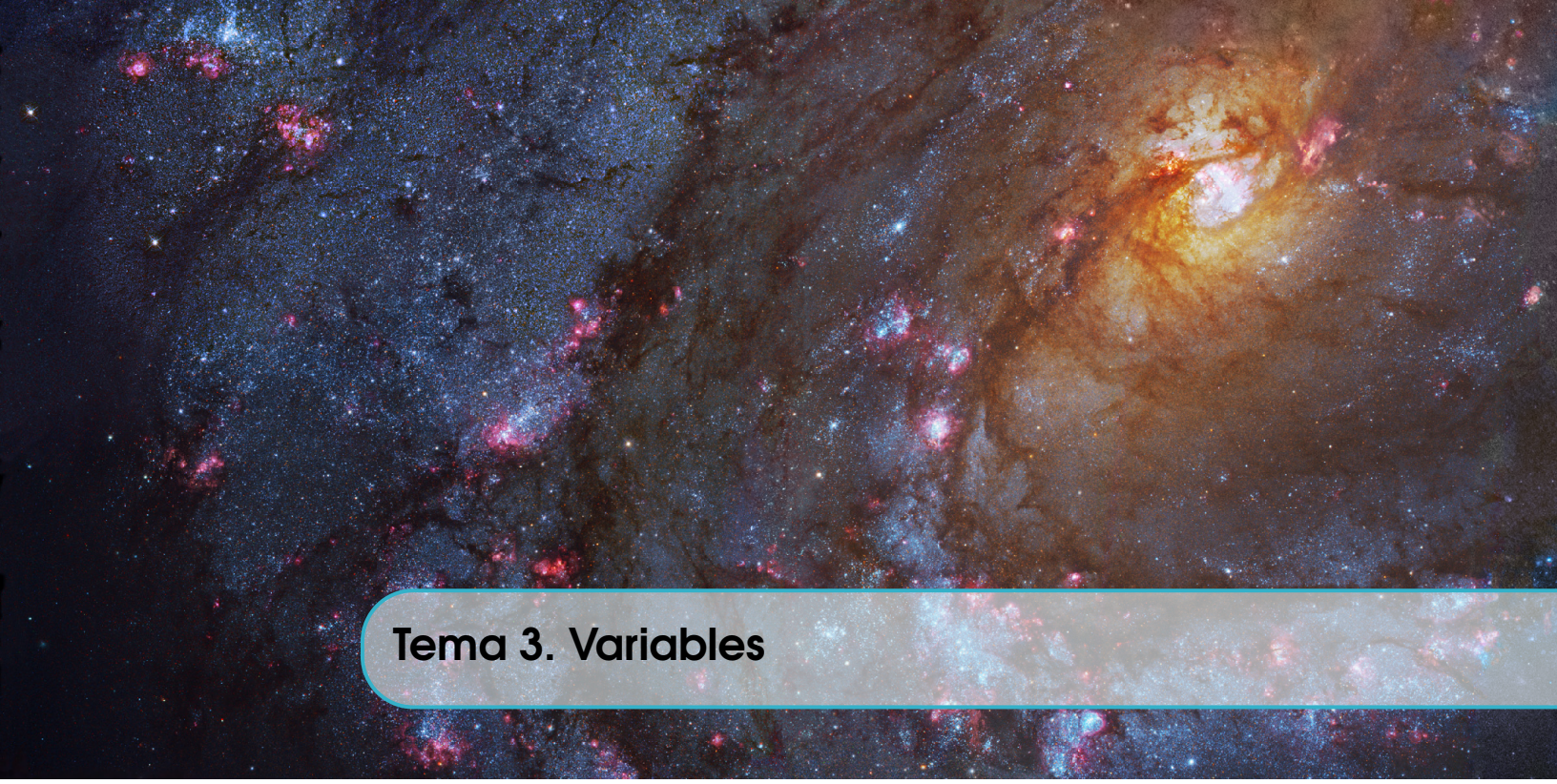
FACULTAD DE MATEMÁTICAS, UNIVERSIDAD DE MURCIA

5ª edición, enero de 2024

Índice general

Parte I

Ejercicios



Tema 3. Variables

3.1 Tema 3: Ejercicios para clase de teoría

3.1 Selecciona los tipos de datos que utilizarías para representar lo siguiente:

- Coordenadas en el espacio
- Un número complejo
- Un polinomio de segundo grado
- Un DNI

3.2 Dadas dos variables a y b de tipo entero, escribe un programa que intercambie sus valores.

3.3 Escribe un programa que muestre por pantalla el resultado de la división cuyo dividendo es el resultado de sumar tres veces la multiplicación de dos valores a y b, y cuyo divisor es el resultado de multiplicar tres veces la suma de dos valores a y b.

3.4 Escribe un programa que, dada una cantidad de dinero en euros, la convierta a dólares americanos. El programa debe mostrar por pantalla ambas cantidades.

3.5 Escribe un programa que, dadas 3 notas aleatorias entre 0 y 10 correspondientes a 3 asignaturas, respectivamente, muestre las tres notas por pantalla y calcule y muestre por pantalla la media de las tres notas.

3.2 Tema 3: Ejercicios para clase de prácticas

3.6 Evalúa y determina el tipo de las siguientes expresiones:
- 3.0+5/4-2
- 2*PI*r (Siendo PI una constante de tipo double con valor igual a 3.14 y r una variable de tipo float que en el momento de realizar la evaluación vale 5.0)
- 'b' > 'z' || true

3.7 Escribe un programa que, dada una temperatura en grados Celsius, la convierta a grados Fahrenheit. El programa debe mostrar por pantalla ambas temperaturas.

3.8 Escribe un programa que, dado un precio P, un descuento d y una propina p, muestre los tres valores y calcule y muestre el valor final v pagado por un producto. Haz dos versiones del programa: una en la que el descuento sea un valor absoluto y otra en la que el descuento sea un porcentaje.

3.9 Escribe un programa que, dado el lado l de un triángulo equilátero, calcule su área. Nota: En Processing, la raíz cuadrada de x se calcula como `sqrt(x)`.

3.10 Escribe un programa que, dada una altura en metros y un peso en kilogramos, muestre por pantalla ambos valores y calcule y muestre el índice de masa corporal a partir de éstos.

3.3 Tema 3: Ejercicios para casa

3.11 Declara las variables, utilizando un identificador representativo, para almacenar los siguientes tipos de valores:
- La altura de una torre en metros.
- El número de profesores en un instituto.
- La letra del piso de una vivienda.
- Indicar si un alumno se ha presentado al examen.
- Un refrán.

3.12 Escribe un programa que genere un número aleatorio en cierto intervalo $[min, max]$ con 3 números decimales.

3.13 Dados 2 puntos del plano $P1 = (x_1, y_1)$ y $P2 = (x_2, y_2)$, donde $x_1 < x_2$ e $y_1 < y_2$, se puede dibujar un triángulo rectángulo cuya hipotenusa sean esos dos puntos: ¡dibújalo en papel! Según tu dibujo, haz un programa que calcule y muestre el perímetro y área del triángulo.

3.14 Escribe un programa que, dada una temperatura en grados Fahrenheit, la convierta a grados Celsius.

3.15 Escribe un programa que, dados los tres lados de un triángulo, calcule su área utilizando la fórmula de Herón.

3.16 Escribe un programa que intercambie los valores de 3 variables. Muestra su contenido antes y después del intercambio. La primera debe almacenar el contenido de la tercera, la segunda debe almacenar el contenido de la primera y la tercera la de la segunda. Sólo se permite el uso de una única variable auxiliar. Haz la tabla de seguimiento usando la ventana de depuración de Processing.

3.17 Realiza los siguientes ejercicios:
- Haz un programa que calcule el volumen de un cilindro y las superficies del mismo a partir de su radio y altura.
- Completa el programa para determinar cuántos metros cuadrados necesitas de hojalata para hacer 100 botes de conserva de 30cm de alto y 10cm de ancho.

Tema 4. Programación estructurada

4.1 Tema 4: Ejercicios para clase de teoría

4.1.1 Estructuras condicionales

4.1 Haz un programa que, dada una temperatura en grados Celsius, si dicha temperatura es mayor que 20, muestre el mensaje "*Se está a gustico aquí*". En caso contrario, mostrar el mensaje "*Parece que va a refrescar; ponte una chaquetica*". Reescribe el programa utilizando solamente una vez la función `print()`.

4.2 Haz un programa que indique qué calificación tiene un alumno dada su nota numérica según el siguiente criterio:

- Suspenso, si la nota es inferior a 5.
- Aprobado, si la nota está en el intervalo $[5,7)$.
- Notable, si la nota está en el intervalo $[7,9)$.
- Sobresaliente, si la nota está en el intervalo $[9,10)$.
- Matrícula de Honor, si la nota es igual a 10.

4.3 Haz un programa que, dado un número natural n, indique si éste es par o impar.

4.1.2 Bucles

4.4 Suma de los primeros n números naturales.

4.5 Suma todos los números enteros entre dos números a y b dados. Repite el ejercicio usando un bucle `while` y un bucle `for`.

4.6 Calcula la media de todos los números enteros comprendidos entre dos números a y b dados.

4.7 Genera números aleatorios mientras que no se encuentre un múltiplo de N (para $N = 5$). Cuando se encuentre dicho múltiplo, muéstralo e indica cuántos números aleatorios se han generado.

4.8 Calcula la potencia a^b usando solamente multiplicaciones.

4.1.3 Estructuras condicionales anidadas

4.9 Haz un programa que muestre el valor mayor y valor menor de tres números dados.

4.1.4 Estructura condicional múltiple

4.10 Sean dos números enteros a y b. Programa una calculadora con cuatro opciones:

1. Sumar $a+b$
2. Restar $a-b$
3. Multiplicar $a \cdot b$
4. Dividir $\frac{a}{b}$, solamente si $b \neq 0$ (si $b = 0$, mostrar mensaje de error)

Selecciona aleatoriamente el tipo de operación a realizar en cada ejecución del programa.

4.2 Tema 4: Ejercicios para clase de prácticas

4.2.1 Estructuras condicionales

4.11 Dado un peso en kilogramos, una altura en metros, y el género de una persona, calcular el índice de masa corporal y, en función de éste, indicar si hay bajo peso, peso normal, sobrepeso u obesidad.

4.12 Haz un programa que, dado un número entre 1 y 12, representando un mes del año, indique el nombre del mes y el número de días que tiene dicho mes. Sólo se permiten 3 `print()` para mostrar el número de días.

4.2.2 Bucles

4.13 Dado un número natural S, sumar todos los números naturales hasta que dicha suma sea mayor o igual que S.

4.14 Un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos. Haz un programa que, dado un número natural x, indique si éste es perfecto o no.

4.15 Haz un programa que muestre todos los números enteros dentro del intervalo $[a,b]$, pero en orden inverso.

4.2.3 Estructuras condicionales anidadas

4.16 Dadas dos fechas (día, mes y año), indicar cuál de ellas es anterior y cuál es posterior.

4.2.4 Estructura condicional múltiple

4.17 Dado el radio de un círculo, calcular su diámetro, perímetro o área con un menú.

4.3 Tema 4: Ejercicios para casa

4.18 Haz un programa que determine si dos puntos coinciden en alguna coordenada.

4.19 Haz un programa que determine si un número x está dentro de un intervalo $[min, max]$ o no.

4.20 El teorema de la desigualdad del triángulo establece que la suma de dos lados del triángulo siempre es mayor que la medida del tercer lado. Si esto resulta ser verdad para todas las tres combinaciones de las sumas, entonces se tiene un triángulo. Haz un programa que, dados tres lados, determine si forman un triángulo o no utilizando el teorema de la desigualdad.

4.21 Determinar si el producto de dos números naturales aleatorios a y b es múltiplo de N, para $N = 7$.

4.22 Dados dos números aleatorios *min* y *max*, garantiza que $min < max$.

4.23 Dados dos círculos determinados por sus respectivos centros y radios, determina si dichos círculos se solapan o no.

4.24 Calcula el inverso de la suma de dos números, mostrando un error si se divide por cero.

4.25 Aplica un descuento del 25 % a un amigo y de un 10 % a los demás.

4.26 Contar el número de divisores que tiene un número natural.

4.27 Un número abundante o número excesivo es un número x para el cual $\sigma(x) > 2x$, donde $\sigma(x)$ es la suma de todos los divisores positivos de x, incluido el propio x. Haz un programa que, dado un número natura x, indique si éste es abundante o no.

4.28 Para determinar la letra del DNI se procede como sigue:

- Se divide el número del DNI entre 23 (lo que da un resto entre 0 y 22).
- A cada resto se le asigna una letra en el siguiente orden, correspondiéndole al resto 0 la letra T y al resto 22 la letra E.

$$T, R, W, A, G, M, Y, F, P, D, X, B, N, J, Z, S, Q, V, H, L, C, K, E$$

Determina la letra de tu DNI y de 4 compañeros más.

4.29 Dados 3 valores numéricos almacenados en las variables a, b y c, ordena los valores de tal forma que al finalizar el proceso se cumpla que $a < b < c$. Imprime el mensaje correspondiente junto con los valores de a, b y c antes y después de la ordenación.

4.30 Como norma general, el límite de velocidad en España para automóviles y motos, dependiendo del tipo de vía, es el siguiente:

- En vías urbanas con un único carril por sentido de circulación, 30 Km/h
- En vías urbanas con más de un carril por sentido de circulación, 50 Km/h
- En carreteras convencionales, 90 Km/h
- En autovías y autopistas, 120 Km/h

Dado un tipo de vía aleatorio y dada una velocidad aleatoria expresada en m/s, indicar si hay multa o no.

En caso de que haya multa, la base son 200 Euros, pero la cantidad final se calcula proporcionalmente al porcentaje en que se supera el límite de velocidad[1]. Por ejemplo, si la velocidad real supera en un 20 % el límite, entonces la multa será de 300 Euros (250 euros de base, más un 20 %).

4.31 En un videojuego, un jugador gana puntos comiendo caramelos con forma circular. Los puntos que da cada caramelo dependen de su color y su área. Si el caramelo es rojo, da 200 puntos, si es verde o su área es mayor de 20, entonces recibe 150 puntos, si es amarillo y su área es

[1] Esta forma de calcular las multas es no se corresponde con la realidad, sino que sirve únicamente para realizar el ejercicio de programación.

menor que 20, entonces recibe 100 puntos. En cualquier otro caso, recibe 50 puntos. Haz un programa que genere un caramelo aleatorio de un determinado color (“Rojo”, “Amarillo” o “Verde”) con un radio aleatorio entre 2 y 3, e indique cuántos puntos ganaría el jugador si se lo comiese.

4.32 El factorial de un número n se calcula como $n! = n \cdot (n-1) \cdot \ldots \cdot 1$ (por ejemplo, $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$). Haz un programa que calcule el factorial de un número natural dado n.

Tema 5. Programación modular

5.1 Tema 5: Ejercicios para clase de teoría

5.1 Define una función que calcule la suma de dos números enteros. Haz un programa que invoque a dicha función con dos sumandos aleatorios entre 1 y 50.

5.2 Define una función que, dado un número x y un intervalo $[a,b]$, indique si $x \in [a,b]$ o no. Prueba tu función N veces (con $N = 10$) con parámetros aleatorios.

5.3 Define una función que, dado un número natural, indica si es primo o no. Prueba N (con $N = 20$) números aleatorios entre 2 y 100 y muestra un mensaje diciendo si son números primos o no.

5.4 Define una función que, dado un nombre, una edad, el sexo y una dirección de correo electrónico, muestre un mensaje con toda la información asociada al perfil de esa persona. Para probar tu función, pide todos estos parámetros al usuario.

5.5 Define una función que calcule el área de un círculo a partir de su radio. A partir de dicha función, define otra función que calcule la superficie de un cilindro dadas su altura y su radio. Prueba esta última función para un radio y una altura aleatorios.

5.6 Define una función que muestre la tabla de multiplicar del número entero x, para los primeros n productos. Pide al usuario los valores de x y n.

5.7 Define una función que indique si un número dado n es múltiplo o no de otro dado m. Haz un programa que pida al usuario el valor de m y que luego genere números aleatorios para n en el rango $[0, 20 \cdot m]$ hasta encontrar N (con $N = 10$) múltiplos de m.

5.8 Define una función que, dado un entero n, indique el número de cifras que lo componen. Prueba tu función con N (para $N = 20$) números aleatorios entre 1 y 10^4.

5.9 Dada una variable `s` de tipo `String`, es posible conocer la longitud de la cadena de la siguiente forma:

```
1 String s = "Cadena";
2 int longitud = s.length(); //La variable longitud almacenará un 6
```

Por otra parte, es posible conocer el carácter que ocupa la posición i-ésima, de la siguiente forma:

```
1 String s = "Cadena";
2 char caracter = s.charAt(0); //La variable caracter almacenará una 'C'
3 caracter = s.charAt(4); //La variable caracter almacenará ahora una 'n'
```

Sabiendo esto, haz una función que, dada una cadena de caracteres, calcule y devuelva el número de vocales que contiene. Prueba tu función con varias cadenas de caracteres que representen un refrán a tu elección.

5.2 Tema 5: Ejercicios para clase de prácticas

5.10 Define una función que, usando la función `random(0,1)` (que devuelve un número aleatorio en el intervalo $[0,1)$), devuelva un número aleatorio entre dos enteros a y b dados. Prueba a ejecutar tu función N veces (con $N = 20$) para $a = 30$ y $b = 50$.

5.11 Define una función que calcule la distancia euclídea entre dos puntos dados (x_1, y_1), (x_2, y_2). Prueba dicha función generando dos puntos aleatorios con la función del ejercicio **5.10**.

5.12 Define una función que, dados dos círculos (definidos por sus respectivos centros y radios), indique si dichos círculos se solapan o no. Prueba tu función con círculos aleatorios cuyas coordenadas de sus centros tomen valores en el intervalo $[-100, 100]$ y cuyos radios tomen valores en el intervalo $[1, 10]$ hasta encontrar N (con $N = 25$) círculos que sí se solapen.

5.13 Define una función que muestra por pantalla los divisores de un número natural n dado. Prueba tu función N veces (con $N = 15$) para un valor n aleatorio entre 2 y 200.

5.14 Dos números amigos son dos números enteros positivos a y b tales que la suma de los divisores propios de uno es igual al otro número y viceversa. Define una función que determine si dos números son amigos o no. A partir de la función anterior, haz las siguientes funciones:

- Una función que busque aleatoriamente N (con $N = 5$) pares de números amigos en el rango $[A, B] = [1, 500]$.
- Una función que busque sistemáticamente todos los pares de números amigos en el rango $[A, B] = [1, 500]$.

5.15 Haz una función que, dado un número natural n, devuelva otro número que sea igual a la suma de las cifras de n.

5.16 Un palíndromo[1] es una palabra o frase que se lee igual en un sentido que en otro (por ejemplo, "Yo hago yoga hoy"). Haz una función que, dada una cadena de caracteres, indique si se trata de un palíndromo o no.

5.17 Haz una función que construya y devuelva una cadena de caracteres de longitud dada n. Cada carácter de la cadena será, o bien un paréntesis abierto '(', o bien un paréntesis cerrado ')', con probabilidad equiprobable.

[1] https://es.wikipedia.org/wiki/PalÃŋndromo

5.3 Tema 5: Ejercicios para casa

5.18 Haz una función que, dados tres puntos (x_i, y_i) (con $i \in \{1,2,3\}$), calcule y devuelva el área del triángulo que conforman. En caso de que esos tres puntos no conformen un triángulo, según el Teorema de la desigualdad, la función devolverá -1 como "código de error".

5.19 Haz una función que, dado un número natural n, devuelva otro número que sea igual a la suma de las cifras de n que ocupan una posición par menos la suma de las cifras que ocupan una posición impar (las unidades ocuparían la posición 1, las decenas la posición 2, etc.). Por ejemplo, para el caso de 28.075.413, tendríamos que:

$$28{,}075{,}413 \rightarrow -14 = (1+5+0+2) - (3+4+7+8)$$

5.20 El último Teorema de Fermat dice que si n es un número entero mayor o igual que 3, entonces no existen números enteros positivos x, y y z, tales que se cumpla la igualdad:

$$x^n + y^n = z^n$$

Haz una función que dados dos números naturales n y max, compruebe si se cumple o no el Teorema de Fermat $\forall x, y, z \in [1, max]$. Prueba la función con $n \in \{3,4\}$ y $max \in \{25, 50, 100\}$.

5.21 Haz una función que, dado un número natural n, calcule su factorial $n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$. Prueba tu función con los primeros 6 números naturales.

5.22 El coeficiente binomial[2] $\binom{n}{k}$ se calcula como

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Haz una función que, a partir de la función del ejercicio **5.21**, calcule el coeficiente binomial $\binom{n}{k}$.

5.23 El Teorema del binomio[3] establece que cualquier potencia de un binomio $x+y$ puede ser expandida en una suma de la forma:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

Sabiendo esto, haz una función que, haciendo uso de la función del ejercicio **5.22**, y dados los números reales x, y y el número natural n, calcule la potencia $(x+y)^n$. Prueba tu función con valores aleatorios de $x, y \in [1, 10]$ y $n \in \{2,3,4,5\}$.

5.24 El número áureo (también llamado número de oro, número de Dios, razón extrema y media, razón áurea, razón dorada, media áurea, proporción áurea y divina proporción) es un número irracional, representado por la letra griega ϕ (phi) (en minúscula) o Φ (Phi) (en mayúscula) en honor al escultor griego Fidias.
Su valor numérico es:

$$\phi = \frac{1+\sqrt{5}}{2}$$

Por otra parte, se dice que dos números naturales a y b están en proporción áurea si se cumple que:

$$\frac{a+b}{a} = \frac{a}{b}$$

[2] https://es.wikipedia.org/wiki/Coeficiente_binomial
[3] https://es.wikipedia.org/wiki/Teorema_del_binomio

Haz una función que, dados dos números naturales a y b, determine si están en proporción áurea o no. A partir de esta función, haz otra función que busque todos los pares de números a y b que están en proporción áurea dentro del intervalo $[min, max]$.

5.25 Dada una cadena aleatoria de paréntesis abiertos y cerrados generada con la función desarrollada en el ejercicio **5.17**, haz una función que cuente cuántos paréntesis están correctamente cerrados. Por ejemplo, la cadena ‘‘)(()()(’’ tiene 2 paréntesis correctamente cerrados, mientras que la cadena ‘‘(()(()())()()’’ tiene 6.

Tema 6. Datos estructurados

6.1 Tema 6: Ejercicios para clase de teoría

6.1 Sobre una lista de naturales (enteros positivos) construir las siguientes funciones:

- Crear una lista de longitud n con valores aleatorios entre un mínimo y un máximo.
- Crear una lista aleatoria de longitud n con los n primeros (y distintos) números naturales.
- Crear una copia de la lista.
- Limpiar todas las posiciones de la lista, asignándoles un valor dado.
- Indicar si un valor dado está contenido en la lista o no.
- Modificar los valores pares de una lista para que valgan la mitad.
- Modificar cada término según la expresión $a_i = a_{i-1} + a_i + a_{i+1}$.
- Mostrar los valores de una lista de naturales.

6.2 Construye cada una de las funciones cuyo propósito se indica a continuación:

- Generar matrices aleatorias para rangos dados.
- Calcular la suma de dos matrices, si es posible.
- Modificar los valores de una matriz multiplicando cada elemento por una constante dada.
- Indicar si dos matrices son iguales o no.
- Generar y devolver una representación de tipo String de la matriz
- Mostrar los valores de una matriz.

6.3 Crea un registro para representar a un Alumno con, al menos los siguientes campos: nombre, apellidos, fecha de nacimiento, sexo, email.
Haz un programa que represente 3 alumnos y muestra su información por pantalla.

6.4 Crea un registro para representar una asignatura con, al menos, los siguientes campos: nombre, créditos, cuatrimestre y calificación (de 0 a 10).
Amplía el ejercicio **6.3** para que cualquier alumno tenga 3 asignaturas.

6.2 Tema 6: Ejercicios para clase de prácticas

6.5 Dado un array de enteros, implementar las siguientes funciones:
- Calcular la media aritmética de todos sus elementos
- Calcular la desviación típica.

6.6 Dados dos vectores $\overrightarrow{u}$ y $\overrightarrow{v}$, implementar las siguientes funciones:
- Calcular el módulo de un vector
- Sumar ambos vectores
- Restar ambos vectores
- Multiplicar un vector por un escalar
- Producto escalar de dos vectores

6.7 Dada una matriz A, implementar las siguientes funciones:
- Calcular la transpuesta A^t
- Indicar si es cuadrada
- Indicar si es simétrica

Haz un programa para mostrar que se cumplen las siguientes propiedades:
- $(A^t)^t = A$ (involutiva)
- $(A+B)^t = A^t + B^t$ (distributiva)
- $(c \cdot A)^t = c \cdot A^t$

6.8 Dada una matriz A, implementar las siguientes funciones:
- Dado un índice i de una fila, eliminar la fila correspondiente de la matriz A.
- Dado un índice j de una columna, eliminar la columna correspondiente de la matriz A.
- Dado un índice i de una fila y una nueva fila, añadir esa nueva fila en el índice dado de la matriz A.
- Dado un índice j de una columna y una nueva columna, añadir esa nueva columna en el índice dado de la matriz A.
- Desplazar las filas de la matriz A hacia abajo, de forma circular, un número dado de posiciones.
- Desplazar las columnas de la matriz A hacia la derecha, de forma circular, un número dado de posiciones.

Nota: Todas las operaciones deberán realizarse sin modificar la matriz original A.

6.9 A partir del ejercicio **6.4**, crea un registro para representar una Fecha y utilízalo dentro del registro de Alumno. Crea las siguientes funciones:
- Dadas dos fechas, calcula la diferencia entre ambas expresada en días. Asume meses de 30 días y sin años bisiestos.
- Dado un número de días, calcular su equivalente en número de años, meses y días.
- Dado un alumno, calcular su edad. Añade este campo al Alumno.
- Generaliza para que un Alumno pueda tener de 1 a n Asignaturas.
- Añade un campo con la nota media de todas las asignaturas de un alumno.

6.10 Crea un registro para representar un punto $p = (x, y)$ y otro registro para representar un círculo c, el cual viene determinado por su centro $p = (p_1, p_2)$ y un radio $r \in \mathbb{R}$. A partir de dichos registros, crea las siguientes funciones:
- Obtener un punto aleatorio cuyas coordenadas toman valores dentro de un rango $[min, max]$.
- Dados dos puntos p_1 y p_2, calcular la distancia euclídea entre ambos.
- Dados dos puntos p_1 y p_2, calcular el punto medio p_m.

- Obtener un círculo aleatorio donde las coordenadas de su centro toman valores dentro de un rango $[min_c, max_c]$ y cuyo radio toma valores dentro de un rango $[min_r, max_r]$
- Dados dos círculos c_1 y c_2, indicar si se solapan o no.

6.3 Tema 6: Ejercicios para casa

6.11 Dado un array A de enteros, implementar las siguientes funciones:
- Devolver otro array B en orden inverso.
- Dado otro array B, devolver la unión de ambos $C = A \cup B$
- Dado otro array B, devolver la intersección de ambos $C = A \cap B$

6.12 Dado un número natural obtén:
- La lista de divisores propios de dicho número.
- La lista de factores primos distintos de dicho número.

6.13 Obtener la lista de números primos que están dentro de un rango de números naturales.

6.14 Obtener la lista de k números primos que son inferiores a un número dado. Justifica qué hacer cuando no se alcancen los k.

6.15 Dada una matriz de $n \times m$ valores reales, construye un función que los retorna en un array1D con todos los valores de la matriz por filas.

Por ejemplo, la transformación de $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times 3}$ es $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}_{1\times 6}$

6.16 A partir del ejercicio **6.9**, crea el registro Clase con al menos los siguientes campos: curso, conjunto de alumnos.

Dada una Clase, Haz las siguientes funciones:
- Devolver el alumno con la máxima nota media de todas sus asignaturas.
- Dada una clase y el nombre de una asignatura, devolver el alumno con la máxima nota para dicha asignatura.
- Dada una clase y el nombre de una asignatura, devolver las notas de todos los alumnos para esa asignatura.
- Devolver la media de edad de la clase.

Crea una Clase con 5 Alumnos, cada uno de ellos con 3 Asignaturas y prueba las funciones anteriores.

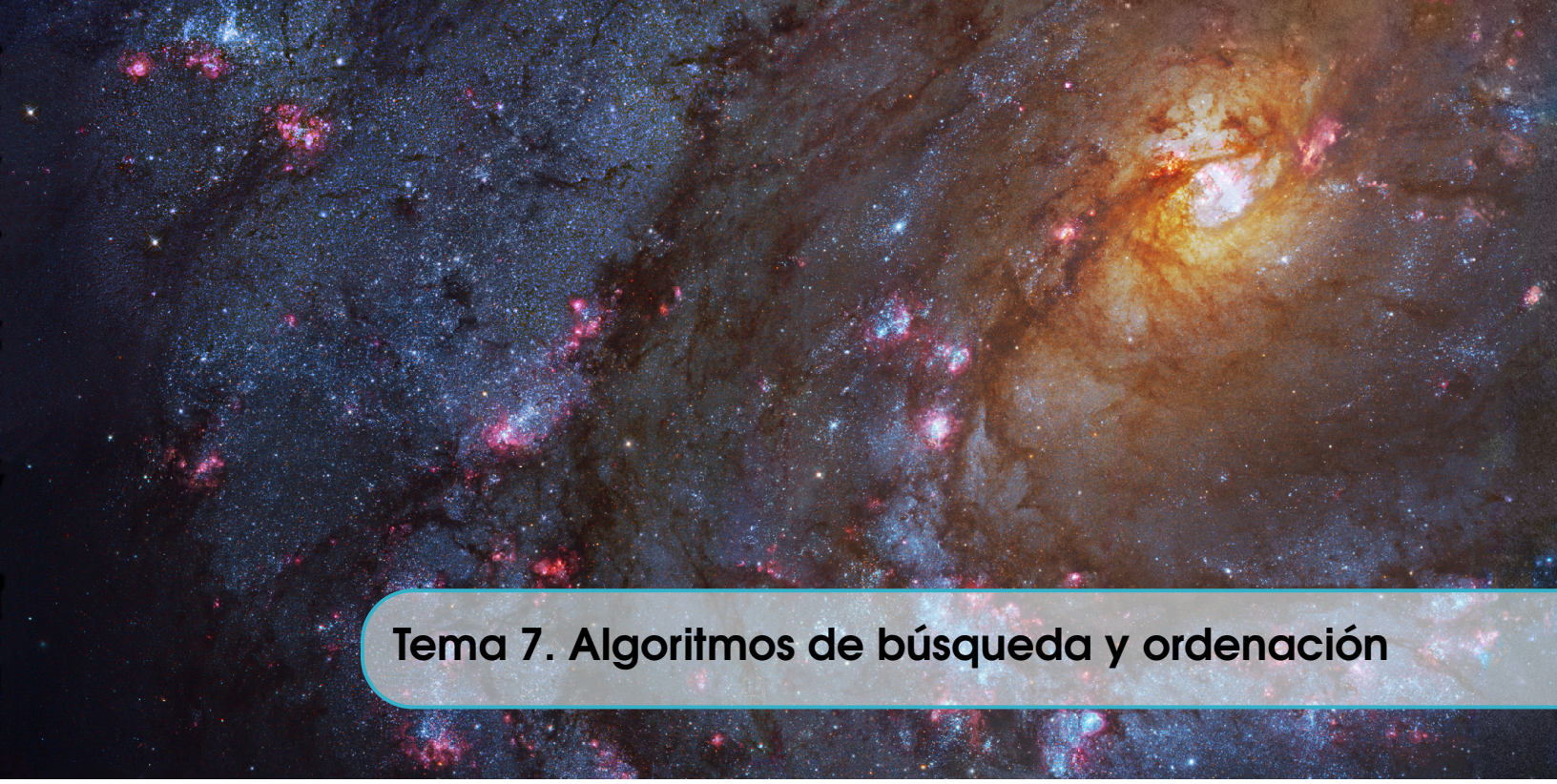
6.17 Dada una lista, devolver otra lista con los valores normalizado al rango $[0,1]$. Por lo tanto, el valor mínimo de la lista devuelta será 0 y el máximo 1.

6.18 Dada una lista de longitud n con números enteros positivos entre 0 y $n-1$, devolver otra lista que contenga el número de repeticiones de cada número. Para ello, en la posición i-ésima de la lista devuelta debe indicarse el número de repeticiones del número i en la lista original.

6.19 Dado un array de dos dimensiones donde cada celda contiene un número entero, crea un programa que encuentre la fila cuya suma de sus valores sea mayor.

6.20 Calcula el determinante de una matriz de 3×3.

6.21 Vamos a representar el mapa de un terreno mediante un array de dos dimensiones, siendo valor de cada casilla la elevación del terreno en esas coordenadas. Se desea realizar una operación de suavizado para obtener una versión más suave del terreno. Esto ayuda a eliminar las irregularidades en la elevación del terreno y a obtener una versión uniforme del mismo. Una forma de suavizar el terreno es promediar el valor de cada punto con el de sus ocho vecinos inmediatos, que representan los puntos en las direcciones horizontal, vertical y diagonal. El resultado de la operación de suavizado es un nuevo array de dos dimensiones donde cada casilla contenga el promedio de la casilla original y de sus vecinos que estén a una casilla de distancia. Ten en cuenta que las casillas situadas en los bordes tienen menos vecinos con los que promediar.



Tema 7. Algoritmos de búsqueda y ordenación

7.1 Tema 7: Ejercicios para clase de teoría

7.1 Sobre una lista de naturales (enteros positivos) construir las siguientes funciones:

- Indicar si un número dado se encuentra o no contenido en la lista.
- Contar el número de veces que un número dado se encuentra en la lista.
- Devolver las posiciones que ocupa un número dado en la lista.

7.2 Sobre una matriz de números naturales, construye cada una de las funciones cuyo propósito se indica a continuación:

- Indicar si un número dado se encuentra o no contenido en la matriz.
- Contar el número de veces que un número dado se encuentra en la matriz.

7.3 Crea un registro para representar a una posición (i, j) dentro de una matriz. Amplía el ejercicio **7.2** para devolver todas las posiciones que ocupa un número dado en una matriz dada.

7.2 Tema 7: Ejercicios para clase de prácticas

7.4 Dado un array de enteros, implementar las siguientes funciones:

- Ordenar la lista de forma ascendente o descendente, según un parámetro de la función.
- Devolver otro array eliminando los números repetidos (independientemente de que el array esté ordenado o no). Por ejemplo, dado el array $\begin{bmatrix}4 & 2 & 2 & 4 & 5 & 6\end{bmatrix}$ debería devolver $\begin{bmatrix}4 & 2 & 5 & 6\end{bmatrix}$.
- Devolver otro array eliminando los números repetidos y que además sus elementos estén ordenados de forma ascendente o descendente, según un parámetro de la función.

7.5 A partir del ejercicio **6.9**, ordenar en orden ascendente las asignaturas de cada alumno en función de su calificación.

7.3 Tema 7: Ejercicios para casa

7.6 A partir del ejercicio **6.16**, crea las siguientes funciones:

- Ordenar todos los alumnos de una clase en función de su nota media.
- Ordenar todos los alumnos de una clase en función de la nota de una asignatura dada.

7.7 Construye las siguientes funciones:

- La que indique cuántas veces aparece cada dígito (0-9) en un determinado número natural.
- La que indique cuántas veces aparece cada dígito en una lista de números naturales.
- Usa una tercera función para mostrar los resultados.

7.8 Dada una cadena de caracteres que representa una secuencia de números naturales separados por espacios (por ejemplo, ‘‘`432 65 854 23 1234`’’), construye las siguientes funciones:

- La que devuelve una lista de cadenas de caracteres, representando cada uno de los números naturales contenidos en la cadena original.
- La que dada una cadena que representa un número natural, la convierte al número en sí mismo. Nota: no se permite el uso de la función `int()` de Processing.
- La que dado un número natural, lo convierte en su representación correspondiente como cadena de caracteres. Nota: no se permiten funciones especiales de Processing, ni “trucos” como concatenar directamente el número dado con la cadena vacía o similar; se espera hacer uso de bucles.
- La que dada la cadena original de secuencia de números naturales, devuelve otra cadena de caracteres con esos mismos números, pero ordenados en orden ascendente. Por ejemplo, para el caso de la cadena ‘‘`432 65 854 23 1234`’’, debería devolver ‘‘`23 65 432 854 1234`’’.

Tema 8. Recursividad

8.1 Tema 8: Ejercicios para clase de teoría

8.1 Calcula de forma recursiva la suma de los n-primeros naturales.

8.2 Dada una lista de números enteros, contar el número de veces que aparece un número dado en dicha lista, haciendo uso de la recursividad.

8.3 Todo número n es primo si no hay divisor entre el 2 y $n/2$. Dado un número n indica, usando recursividad, si es primo o no.

8.2 Tema 8: Ejercicios para clase de prácticas

8.4 Calcula de forma recursiva el factorial de un número n.

8.5 La sucesión de Fibonacci es:

$$1, 1, 2, 3, 5, 8, 13 \ldots$$

Usando recursividad, indica cuál es el i-ésimo término de la sucesión de Fibonacci.

8.3 Tema 8: Ejercicios para casa

8.6 Haz una función recursiva que muestre por pantalla los dígitos de un número n dado.

8.7 Haz una función recursiva que devuelva la suma de los dígitos que componen un número n dado.

8.8 Dada una lista de números, hacer una función recursiva que devuelva la misma lista, pero en orden inverso.

8.9 Dada una lista de números, hacer una función recursiva que devuelva la suma de sus elementos.

8.10 Haz una función recursiva para calcular la potencia de un número a^b.

Tema 10. Programación orientada a objetos

10.1 Tema 10: Ejercicios para clase de teoría

10.1 Crea una clase para representar círculos definidos por su radio y su color. Permitir calcular el área y el perímetro de un círculo.

10.2 A partir del ejercicio **10.1**, crea una clase para representar cilindros, que permitan calcular su superficie y su volumen.

10.3 Crea una clase para representar un punto (x, y) en el eje de coordenadas cartesianas. Permitir calcular la distancia euclídea entre dos puntos.

10.4 Amplía el ejercicio **10.1** para permitir saber si dos círculos se solapan o no.

10.2 Tema 10: Ejercicios para clase de prácticas

10.5 Una Pila es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se apilan y desapilan elementos. Los elementos se apilan uno detrás de otro y, a la hora de desapilar, se elimina el último elemento que haya sido apilado.
Construye una clase que represente una Pila de números naturales de un tamaño máximo dado, con las operaciones para apilar, desapilar, indicar si la pila está vacía y mostrar los elementos de la pila por pantalla.

10.6 A partir del ejercicio **10.3**, construye una clase para representar una recta en el plano definida a partir de dos puntos A y B, con $A \neq B$. Permitir las operaciones para saber si dos rectas son paralelas o no, para obtener el punto de intersección (en caso de que no sean paralelas), para conocer la distancia entre dos rectas (en caso de que sean paralelas) y para conocer la distancia de un punto a una recta.

10.3 Tema 10: Ejercicios para casa

10.7 Una Cola es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se añaden y se eliminan elementos. Los elementos se añaden uno detrás de otro y, a la hora de eliminar un elemento, se elimina más antiguo, y el resto de elementos avanzan una posición en la cola.
Construye una clase que represente una Cola de números naturales de un tamaño máximo dado, con las operaciones de añadir elemento, eliminar elemento, indicar si la cola está vacía y mostrar los elementos de la cola por pantalla.

10.8 Modifica el código del ejercicio **10.7** para que la cola no tenga un tamaño máximo.

10.9 Modifica el código del ejercicio **7.6** para utilizar clases, objetos y métodos en lugar de registros.

Parte II

Ejercicios y soluciones

Tema 3. Variables (Soluciones)

3.1 Tema 3: Ejercicios para clase de teoría

3.1 Selecciona los tipos de datos que utilizarías para representar lo siguiente:

- Coordenadas en el espacio
- Un número complejo
- Un polinomio de segundo grado
- Un DNI

```
1  //Coordenadas (3.0, 2.0) en el espacio
2  float coordenadaX = 3.0;
3  float coordenadaY = 2.0;

5  //Número complejo
6  float parteReal = 4.3;
7  float parteImaginaria = 5.5;

9  //Polinomio de segundo grado a*x^2+b*x+c
10 float coeficienteX2 = -38;
11 float coeficienteX = 26;
12 float terminoIndependiente = -7;

14 //DNI
15 //Opción 1
16 int dni = 12345678;
17 char letraDni = 'Z';

19 //Opción 2
20 String nif = "12345678Z";
```

3.2 Dadas dos variables a y b de tipo entero, escribe un programa que intercambie sus valores.

```
1  int a = 5;
2  int b = 2;
```

```
println("a = "+a);
println("b = "+b+"\n");

int aux = a;
a = b;
b = aux;

println("a = "+a);
println("b = "+b);
```

3.3 Escribe un programa que muestre por pantalla el resultado de la división cuyo dividendo es el resultado de sumar tres veces la multiplicación de dos valores a y b, y cuyo divisor es el resultado de multiplicar tres veces la suma de dos valores a y b.

```
//Dos valores aleatorios a,b ∈ [-10,10)
float a = random(-10, 10);
float b = random(-10, 10);

println("a = "+a);
println("b = "+b);

//Atención a los paréntesis y a la precedencia de los operadores
float dividendo = a*b + a*b + a*b;
float divisor = (a+b) * (a+b) * (a+b);

float division = dividendo/divisor;

print("division = "+division);
```

3.4 Escribe un programa que, dada una cantidad de dinero en euros, la convierta a dólares americanos. El programa debe mostrar por pantalla ambas cantidades.

```
//Definimos una cantidad aleatoria de euros entre 0 y 100
float cantidadEuros = random(0, 100);

//Esta variable almacena el cambio actual de EUR a USD
float cambioEURtoUSD = 1.14;

//Convertimos dicha cantidad en (EUR) a dólares (USD)
float cantidadDolares = cantidadEuros*cambioEURtoUSD;

println("euros = "+cantidadEuros+" EUR");
println("dólares = "+cantidadDolares+" USD");
```

3.5 Escribe un programa que, dadas 3 notas aleatorias entre 0 y 10 correspondientes a 3 asignaturas, respectivamente, muestre las tres notas por pantalla y calcule y muestre por pantalla la media de las tres notas.

```
//Nota aleatoria de álgebra entre 0 y 10
float notaAlgebra = random(0, 10);

//Nota aleatoria de cálculo entre 0 y 10
float notaCalculo = random(0, 10);

//Nota aleatoria de lógica entre 0 y 10
float notaLogica = random(0, 10);
```

```
10 //Nota media de las tres asignaturas
11 float notaMedia = (notaAlgebra + notaCalculo + notaLogica)/3.0;

13 println("Nota de Álgebra = "+notaAlgebra);
14 println("Nota de Cálculo = "+notaCalculo);
15 println("Nota de Lógica = "+notaLogica+"\n");

17 println("Nota media = "+notaMedia);
```

3.2 Tema 3: Ejercicios para clase de prácticas

3.6 Evalúa y determina el tipo de las siguientes expresiones:

- 3.0+5/4-2
- 2*PI*r (Siendo PI una constante de tipo double con valor igual a 3.14 y r una variable de tipo float que en el momento de realizar la evaluación vale 5.0)
- 'b' > 'z' || true

```
float expresion1 = 3.0+5/4-2;
println("Expresión 1 = "+expresion1);

final float PI = 3.14;
float r = 5.0;
float expresion2 = 2*PI*r;
println("Expresión 2 = "+expresion2);

boolean expresion3 = 'b' > 'z' || true;
println("Expresión 3 = "+expresion3);
```

3.7 Escribe un programa que, dada una temperatura en grados Celsius, la convierta a grados Fahrenheit. El programa debe mostrar por pantalla ambas temperaturas.

```
//Temperatura en grados Celsius
float gradosCelsius = random(-10, 40);
println("La temperatura en grados Celsius es: "+gradosCelsius+"C");

//Temperatura en grados Fahrenheit
float gradosFahrenheit = (gradosCelsius * 9.0/5.0) + 32;
println("La temperatura en grados Fahrenheit es: "
    +gradosFahrenheit+"F");
```

3.8 Escribe un programa que, dado un precio P, un descuento d y una propina p, muestre los tres valores y calcule y muestre el valor final v pagado por un producto. Haz dos versiones del programa: una en la que el descuento sea un valor absoluto y otra en la que el descuento sea un porcentaje.

```
float precio = random(20,100);
float descuento = random(5,15);
float propina = random(0,10);

//Opción 1
float valorPagado = precio - descuento + propina;
println("Opción 1:");
println("Valor = P - d + p = "+precio+" - "+descuento+" + "+propina+"
    = "+valorPagado+"\n");

//Opción 2
descuento = random(0,1);
valorPagado = precio * (1 - descuento) + propina;
println("Opción 2:");
println("Valor = P * (100% - d) + p = "+precio+" * "
    +(1-descuento)*100+"% + "+propina+" = "+valorPagado+"\n");
```

3.9 Escribe un programa que, dado el lado l de un triángulo equilátero, calcule su área. Nota: En Processing, la raíz cuadrada de x se calcula como `sqrt(x)`.

```
float lado = random(1, 100);

//Por el Teorema de Pitágoras...
float altura = sqrt(lado*lado - (lado/2)*(lado/2));

//Área = (base * altura)/2
float area = (lado * altura)/2;
println("El área del triángulo de lado "+lado+" es igual a "+area);
```

3.10 Escribe un programa que, dada una altura en metros y un peso en kilogramos, muestre por pantalla ambos valores y calcule y muestre el índice de masa corporal a partir de éstos.

```
float altura = random(1.5,1.9);
float peso = random(40, 150);

float indiceMasaCorporal = peso / (altura*altura);
println("Índice de masa corporal (IMC) = Peso(Kg) / Altura(m2) = "
    +peso+" / ("+altura+"*"+altura+") = "+indiceMasaCorporal);
```

3.3 Tema 3: Ejercicios para casa

3.11 Declara las variables, utilizando un identificador representativo, para almacenar los siguientes tipos de valores:

- La altura de una torre en metros.
- El número de profesores en un instituto.
- La letra del piso de una vivienda.
- Indicar si un alumno se ha presentado al examen.
- Un refrán.

```
int alturaTorre = 50;

int numProfesores = 33;

char letraPiso = 'A';

boolean presentadoExamen = false;

String refran = "Más vale pájaro en mano, que ciento volando";
```

3.12 Escribe un programa que genere un número aleatorio en cierto intervalo $[min, max]$ con 3 números decimales.

```
int min = 0;
int max = 100;
println("(min, max) = ("+min+", "+max+")");

float aleatorio = random(min, max);
println("Número aleatorio: "+aleatorio);

float aleatorio3Decimales = ((int)(aleatorio*1000))/1000.0;
println("Número aleatorio con 3 decimales: "+aleatorio3Decimales);
```

3.13 Dados 2 puntos del plano $P1 = (x_1, y_1)$ y $P2 = (x_2, y_2)$, donde $x_1 < x_2$ e $y_1 < y_2$, se puede dibujar un triángulo rectángulo cuya hipotenusa sean esos dos puntos: ¡dibújalo en papel! Según tu dibujo, haz un programa que calcule y muestre el perímetro y área del triángulo.

```
float x1 = random(0,10);
float y1 = random(0,10);
println("P1= (x1, y1) = ("+x1+", "+y1+")");

float x2 = random(10,20);
float y2 = random(10,20);
println("P2= (x2, y2) = ("+x2+", "+y2+")");

//La hipotenusa se calcula como la distancia entre P1 y P2
float hipotenusa = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
float base = x2-x1;
float altura = y2-y1;

float area = (base*altura)/2;
float perimetro = hipotenusa+base+altura;

println("Área = "+area);
println("Perímtetro = "+perimetro);
```

3.14 Escribe un programa que, dada una temperatura en grados Fahrenheit, la convierta a grados Celsius.

```
//Temperatura en grados Fahrenheit
float gradosFahrenheit = random(0, 100);
println("La temperatura en grados Fahrenheit es: "
    +gradosFahrenheit+"F");

//Temperatura en grados Celsius
float gradosCelsius = (gradosFahrenheit - 32) * (5.0/9.0);
println("La temperatura en grados Celsius es: "+gradosCelsius+"C");
```

3.15 Escribe un programa que, dados los tres lados de un triángulo, calcule su área utilizando la fórmula de Herón.

```
//Lados del triángulo
float lado1 = random(1, 20);
float lado2 = random(1, 20);
float lado3 = random(1, 20);
println("Los lados (a, b, c) del triángulo son ("+lado1+", "+lado2+",
    "+lado3+")");

//Calculamos el semiperímetro
float semiperimetro = (lado1+lado2+lado3)/2.0;
println("El semiperímetro s del triángulo es "+semiperimetro);

//Calculamos el área del triángulo según la fórmula de Herón
//¿Por qué en algunas ocasiones el área vale NaN (Not a Number)?
float area = sqrt(semiperimetro *
                  (semiperimetro - lado1) *
                  (semiperimetro - lado2) *
                  (semiperimetro - lado3));
println("El área del triángulo es "+area);
```

3.16 Escribe un programa que intercambie los valores de 3 variables. Muestra su contenido antes y después del intercambio. La primera debe almacenar el contenido de la tercera, la segunda debe almacenar el contenido de la primera y la tercera la de la segunda. Sólo se permite el uso de una única variable auxiliar. Haz la tabla de seguimiento usando la ventana de depuración de Processing.

```
//Tres valores aleatorios entre 1 y 100
float valor1 = random(1, 100);
float valor2 = random(1, 100);
float valor3 = random(1, 100);
println("Los valores (a, b, c) son ("+valor1+", "+valor2+", "
    +valor3+")");

float aux = valor1;
valor1 = valor3;
valor3 = valor2;
valor2 = aux;

println("Los nuevos valores (a, b, c) son ("+valor1+", "+valor2+", "
    +valor3+")");
```

3.17 Realiza los siguientes ejercicios:

- Haz un programa que calcule el volumen de un cilindro y las superficies del mismo a partir de su radio y altura.
- Completa el programa para determinar cuántos metros cuadrados necesitas de hojalata para hacer 100 botes de conserva de 30cm de alto y 10cm de ancho.

```
1  //Radio r y altura h del cilindro aleatorios entre 1 y 50
2  float radio = random(1, 50);
3  float altura = random(1, 50);
4  println("El radio r del cilindro es "+radio);
5  println("La altura h del cilindro es "+altura);

7  //El volumen V de un cilindro se calcula como V = π*r^2*h
8  final float PI = 3.14;
9  float areaBase = PI * radio * radio;
10 float volumen = areaBase * altura;
11 println("El volumen V del cilindro es "+volumen);

13 //El área A de un cilindro se calcula como A = 2*π*r^2+2*π*r*h
14 float areaLateral = 2 * PI * radio * altura;
15 float areaCilindro = 2*areaBase + areaLateral;
16 println("El área A del cilindro es "+areaCilindro+"\n");

18 //Definimos la altura y anchura de los botes de conserva
19 float alturaBote = 30;
20 float anchuraBote = 10;

22 //Calculamos la superficie de un bote de conserva
23 float superficieBaseBote = PI * (anchuraBote/2.0) * (anchuraBote/2.0);
24 float superficieLateralBote = PI * anchuraBote * alturaBote;
25 float superficieBote = 2*superficieBaseBote + superficieLateralBote;
26 println("Un bote de altura "+alturaBote+" cm y anchura "
       +anchuraBote+" cm require una superficie de "+superficieBote+"
       cm2");
27 println("Por lo tanto, 100 botes, requerirán "+(superficieBote*100)+"
       cm2");
```

Tema 4. Programación estructurada (Soluciones)

4.1 Tema 4: Ejercicios para clase de teoría

4.1.1 Estructuras condicionales

4.1 Haz un programa que, dada una temperatura en grados Celsius, si dicha temperatura es mayor que 20, muestre el mensaje "*Se está a gustico aquí*". En caso contrario, mostrar el mensaje "*Parece que va a refrescar; ponte una chaquetica*". Reescribe el programa utilizando solamente una vez la función `print()`.

```
1  //Temperatura aleatoria en grados Celsius entre 0 y 40
2  float temperaturaCelsius = random(0,40);
3  println("Temperatura = "+temperaturaCelsius+" C");

5  String mensaje = "";
6  if (temperaturaCelsius > 20) {
7    //println("Se está a gustico aquí");

9    mensaje = "Se está a gustico aquí";
10 } else {
11   //println("Parece que va a refrescar; ponte una chaquetica");

13   mensaje = "Parece que va a refrescar; ponte una chaquetica";
14 }

16 println(mensaje);
```

4.2 Haz un programa que indique qué calificación tiene un alumno dada su nota numérica según el siguiente criterio:

- Suspenso, si la nota es inferior a 5.
- Aprobado, si la nota está en el intervalo $[5,7)$.
- Notable, si la nota está en el intervalo $[7,9)$.
- Sobresaliente, si la nota está en el intervalo $[9,10)$.
- Matrícula de Honor, si la nota es igual a 10.

```
//Nota numérica entre 0 y 10
float nota = random(0,10);

//Como la función random nunca va a devolver 10, hacemos lo siguiente para aproximar a 10
if (nota > 9.9) {
  nota = 10;
}
println("Nota numérica = "+nota);

String calificacion = "";

if (nota < 5) {
  calificacion = "Suspenso";
} else if ((nota >= 5) && (nota < 7)){
  calificacion = "Aprobado";
} else if ((nota >= 7) && (nota < 9)){
  calificacion = "Notable";
} else if ((nota >= 9) && (nota < 10)){
  calificacion = "Sobresaliente";
} else {
  calificacion = "Matrícula de Honor";
}

println(calificacion);
```

4.3 Haz un programa que, dado un número natural n, indique si éste es par o impar.

```
//Número natural aleatorio entre 0 y 99
int natural = (int)random(0,100);

if (natural%2 == 0) {
  println("El número "+natural+" es par");
} else {
  println("El número "+natural+" es impar");
}
```

4.1.2 Bucles

4.4 Suma de los primeros n números naturales.

```
//Número natural aleatorio entre 1 y 99
int n = (int)random(1,100);
int contador = 1;
int suma = 0;

while (contador <= n) {
  suma = suma + contador; //Equivalente a suma += contador;
  contador = contador + 1; //Equivalente a contador++;
}
println("La suma de los "+n+" primeros números naturales (usando un bucle while) es: "+suma);

suma = 0;
for (int i = 1; i <= n; i++){
  suma = suma + i;
```

```
}
println("La suma de los "+n+" primeros números naturales (usando un
    bucle for) es: "+suma);
```

4.5 Suma todos los números enteros entre dos números a y b dados. Repite el ejercicio usando un bucle `while` y un bucle `for`.

```
//Número natural aleatorio entre 1 y 10
int a = (int)random(1,11);

//Número natural aleatorio entre a+1 y a+10
int b = a + (int)random(1,11);
int contador = a;
int suma = 0;

while (contador <= b) {
  suma = suma + contador; //Equivalente a suma += contador;
  contador = contador + 1; //Equivalente a contador++;
}
println("La suma de todos los números entre "+a+" y "+b+" (usando un
    bucle while) es: "+suma);

suma = 0;
for (int i = a; i <= b; i++){
  suma = suma + i;
}
println("La suma de todos los números entre "+a+" y "+b+" (usando un
    bucle for) es: "+suma);
```

4.6 Calcula la media de todos los números enteros comprendidos entre dos números a y b dados.

```
//Número natural aleatorio entre 1 y 10
int a = (int)random(1,11);

//Número natural aleatorio entre a+1 y a+10
int b = a + (int)random(1,11);

int suma = 0;
for (int i = a; i <= b; i++){
  suma = suma + i;
}

//Nota: debemos hacer el casting a float para que la división sea real y no entera
float media = suma / (float)(b-a+1);
println("La suma de todos los números entre "+a+" y "+b+" es: "+suma);
println("Y la media es: "+media);
```

4.7 Genera números aleatorios mientras que no se encuentre un múltiplo de N (para $N = 5$). Cuando se encuentre dicho múltiplo, muéstralo e indica cuántos números aleatorios se han generado.

```
//Número natural aleatorio entre 1 y 99
int x = (int)random(1,100);
int N = 5;
int contador = 0;

while(x%N != 0){
```

```
  contador++;
  x = (int)random(1,100);
}
println("El número "+x+" es múltiplo de "+N);
println("Hemos necesitado "+contador+" intentos para encontrarlo");
```

4.8 Calcula la potencia a^b usando solamente multiplicaciones.

```
//Números naturales aleatorios entre 1 y 10
int a = (int)random(1,11);
int b = (int)random(1,11);

int potencia = 1;

for (int i = 0; i < b; i++){
  potencia = potencia * a;
}
println("La potencia de "+a+" elevado a "+b+" es igual a "+potencia);
```

4.1.3 Estructuras condicionales anidadas

4.9 Haz un programa que muestre el valor mayor y valor menor de tres números dados.

```
//3 números naturales aleatorios entre 1 y 10
int a = (int)random(1,11);
int b = (int)random(1,11);
int c = (int)random(1,11);

println("a = "+a+" b = "+b+" c = "+c);

if (a < b) {
  if (c < a) {
    println("El valor mayor es "+b);
    println("El valor menor es "+c);
  } else if (b < c) {
    println("El valor mayor es "+c);
    println("El valor menor es "+a);
  } else {
    println("El valor mayor es "+b);
    println("El valor menor es "+a);
  }
} else {
  //b <a
  if (c < b) {
    println("El valor mayor es "+a);
    println("El valor menor es "+c);
  } else if (a < c) {
    println("El valor mayor es "+c);
    println("El valor menor es "+b);
  } else {
    println("El valor mayor es "+a);
    println("El valor menor es "+b);
  }
}
```

4.1.4 Estructura condicional múltiple

4.10 Sean dos números enteros a y b. Programa una calculadora con cuatro opciones:

1. Sumar $a+b$
2. Restar $a-b$
3. Multiplicar $a \cdot b$
4. Dividir $\frac{a}{b}$, solamente si $b \neq 0$ (si $b = 0$, mostrar mensaje de error)

Selecciona aleatoriamente el tipo de operación a realizar en cada ejecución del programa.

```
//2 números naturales aleatorios entre 0 y 10
int a = (int)random(0,11);
int b = (int)random(0,11);

//Operación aleatoria entre 1 y 4
int operacion = (int)random(1,5);

switch(operacion) {
  case 1: //Sumar
    println("a + b = "+a+" + "+b+" = "+(a+b));
    break;

  case 2: //Restar
    println("a - b = "+a+" - "+b+" = "+(a-b));
    break;

  case 3: //Multiplicar
    println("a * b = "+a+" * "+b+" = "+(a*b));
    break;

  case 4: //Dividir
    if (b != 0) {
      println("a / b = "+a+" / "+b+" = "+(a/(float)b));
    } else {
      println("¡Error! No se puede dividir por cero");
    }
    break;
}
```

4.2 Tema 4: Ejercicios para clase de prácticas

4.2.1 Estructuras condicionales

4.11 Dado un peso en kilogramos, una altura en metros, y el género de una persona, calcular el índice de masa corporal y, en función de éste, indicar si hay bajo peso, peso normal, sobrepeso u obesidad.

```
//Peso aleatorio entre 40 y 100 Kg
float peso = random(40,100);

//Altura aleatoria entre 1 y 2 metros
float altura = random(1,2);

//Como programador decido que si genero == true, entonces representa a una mujer. En otro caso, representa a un hombre
boolean genero;
//Con probabilidad equiprobable podrá ser un hombre o una mujer
if (random(0,1) > 0.5){
  genero = true;
  print("Mujer, ");
} else {
  genero = false;
  print("Varón, ");
}

//Calculamos el IMC
float indiceMasaCorporal = peso/(altura*altura);
print(peso+" Kg, "+altura+" m, IMC = "+indiceMasaCorporal+", tiene ");

if (indiceMasaCorporal < 18.5) {
  println("Bajo peso");
} else if (indiceMasaCorporal < 25) {
  println("Peso normal");
} else if (indiceMasaCorporal < 30) {
  println("Sobrepeso");
} else {
  println("Obesidad");
}
```

4.12 Haz un programa que, dado un número entre 1 y 12, representando un mes del año, indique el nombre del mes y el número de días que tiene dicho mes. Sólo se permiten 3 `print()` para mostrar el número de días.

```
//Número aleatorio entre 1 y 12
int mes = (int)random(1,13);

//Nombre del mes
if (mes == 1) {
  print("Enero ");
} else if (mes == 2) {
  print("Febrero ");
} else if (mes == 3) {
  print("Marzo ");
} else if (mes == 4) {
  print("Abril ");
} else if (mes == 5) {
  print("Mayo ");
```

```
} else if (mes == 6) {
  print("Junio ");
} else if (mes == 7) {
  print("Julio ");
} else if (mes == 8) {
  print("Agosto ");
} else if (mes == 9) {
  print("Septiembre ");
} else if (mes == 10) {
  print("Octubre ");
} else if (mes == 11) {
  print("Noviembre ");
} else {
  print("Diciembre ");
}

print("tiene ");

//Número de días
if ((mes == 1) || (mes == 3) ||
    (mes == 5) || (mes == 7) ||
    (mes == 8) || (mes == 10) ||
    (mes == 12)) {
  print("31 días ");
} else if (mes == 2) {
  print("28 días ");
} else {
  print("30 días ");
}
```

4.2.2 Bucles

4.13 Dado un número natural S, sumar todos los números naturales hasta que dicha suma sea mayor o igual que S.

```
//Número natural aleatorio entre 1 y 200
int S = (int)random(1,201);

int n = 1;
int suma = 0;

while(suma < S) {
  suma = suma + n;//Equivalente a suma += n;
  n = n + 1;//Equivalente a n++;
}

println("S = "+S+"; suma = "+suma+"; n = "+n);
```

4.14 Un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos. Haz un programa que, dado un número natural x, indique si éste es perfecto o no.

```
//Número natural aleatorio entre 1 y 30
int x = (int)random(1,31);
int sumaDivisores = 0;
int divisor = 1;
```

```
while(divisor < x) {
  if (x%divisor == 0) {
    sumaDivisores = sumaDivisores + divisor;
  }
  divisor = divisor + 1;
}

if (x == sumaDivisores) {
  println("x = "+x+" es un número perfecto");
} else {
  println("x = "+x+" NO es un número perfecto");
}
```

4.15 Haz un programa que muestre todos los números enteros dentro del intervalo $[a,b]$, pero en orden inverso.

```
//Número natural aleatorio entre 1 y 10
int a = (int)random(1,11);

//Número natural aleatorio entre a+1 y a+10
int b = a + (int)random(1,11);

println("[a, b] = ["+a+", "+b+"]\n");

for(int i = b; i >= a; i--) {
  println(i);
}
```

4.2.3 Estructuras condicionales anidadas

4.16 Dadas dos fechas (día, mes y año), indicar cuál de ellas es anterior y cuál es posterior.

```
//Años aleatorios entre 1950 y 2022
int anyo1 = (int)random(1950,2023);
int anyo2 = (int)random(1950,2023);

//Meses aleatorios entre 1 y 12
int mes1 = (int)random(1,13);
int mes2 = (int)random(1,13);

//Días aleatorios entre 1 y 30
//Nota: no comprobamos que las fechas tengan sentido
int dia1 = (int)random(1,31);
int dia2 = (int)random(1,31);

println("Fecha 1: "+dia1+"/"+mes1+"/"+anyo1);
println("Fecha 2: "+dia2+"/"+mes2+"/"+anyo2);

if (anyo1 < anyo2) {
  println("La fecha 1 es anterior a la fecha 2");
} else if (anyo1 > anyo2) {
  println("La fecha 2 es anterior a la fecha 1");
} else {
  //Mismo año
  if (mes1 < mes2){
```

```
24      println("La fecha 1 es anterior a la fecha 2");
25    } else if (mes1 > mes2){
26      println("La fecha 2 es anterior a la fecha 1");
27    } else {
28      //Mismo año y mismo mes
29      if (dia1 < dia2){
30        println("La fecha 1 es anterior a la fecha 2");
31      } else if (dia2 < dia1){
32        println("La fecha 2 es anterior a la fecha 1");
33      } else {
34        //Mismo año, mismo mes y mismo día
35        println("¡¡La fecha 1 y la fecha 2 son iguales!!");
36      }
37    }
38  }
```

4.2.4 Estructura condicional múltiple

4.17 Dado el radio de un círculo, calcular su diámetro, perímetro o área con un menú.

```
1  //Radio aleatorio entre 1 y 100
2  float radio = random(1,100);
3  println("Radio = "+radio);

5  final float PI = 3.14;

7  //Opción aleatoria entre 1 y 3
8  int opcion = (int)random(1,4);

10 switch(opcion) {
11   case 1://Calcular diámetro
12     float diametro = 2*radio;
13     println("Diámetro = "+diametro);
14     break;
15   case 2://Calcular perímetro
16     float perimetro = 2*PI*radio;
17     println("Perímetro = "+perimetro);
18     break;
19   case 3://Calcular área
20     float area = PI*radio*radio;
21     println("Área = "+area);
22     break;
23 }
```

4.3 Tema 4: Ejercicios para casa

4.18 Haz un programa que determine si dos puntos coinciden en alguna coordenada.

```
//Coordenadas aleatorias
int x1 = (int)random(0,11);
int y1 = (int)random(0,11);
println("P1 = (x1, y1) = ("+x1+", "+y1+")");

int x2 = (int)random(0,11);
int y2 = (int)random(0,11);
println("P2 = (x2, y2) = ("+x2+", "+y2+")");

if ((x1 == x2) || (y1 == y2)){
  println("Los puntos P1 y P2 sí coinciden en alguna coordenada");
} else {
  println("Los puntos P1 y P2 no coinciden en ninguna coordenada");
}
```

4.19 Haz un programa que determine si un número x está dentro de un intervalo $[min, max]$ o no.

```
//Mínimo aleatorio entre 0 y 50
float min = random(0,50);

//Máximo aleatorio entre min+50 y min+100
float max = min+random(50,100);

//Número aleatorio entre -100 y 200
float x = random(-100,200);

if ((x >= min) && (x <= max)){
  println("El número x = "+x+" sí está dentro del intervalo ["+min+",
      "+max+"]");
} else {
  println("El número x = "+x+" no está dentro del intervalo ["+min+",
      "+max+"]");
}
```

4.20 El teorema de la desigualdad del triángulo establece que la suma de dos lados del triángulo siempre es mayor que la medida del tercer lado. Si esto resulta ser verdad para todas las tres combinaciones de las sumas, entonces se tiene un triángulo. Haz un programa que, dados tres lados, determine si forman un triángulo o no utilizando el teorema de la desigualdad.

```
//Lados aleatorios entre 1 y 100
float lado1 = random(1,100);
float lado2 = random(1,100);
float lado3 = random(1,100);
println("Lado 1 = "+lado1+"; Lado 2 = "+lado2+"; Lado 3 = "+lado3);

//Teorema de la desigualdad del triángulo
if ((lado1+lado2 > lado3) &&
    (lado1+lado3 > lado2) &&
    (lado2+lado3 > lado1)){
  println("Los tres lados sí forman un triángulo");
} else {
  println("Los tres lados no forman un triángulo");
}
```

4.21 Determinar si el producto de dos números naturales aleatorios a y b es múltiplo de N, para $N = 7$.

```
//Números naturales aleatorios entre 1 y 20
int a = (int)random(1,21);
int b = (int)random(1,21);
println("a = "+a+"; b = "+b);

int N = 7;

if ((a*b)%N == 0){
  println("El producto a*b = "+(a*b)+" sí es múltiplo de N = "+N);
} else {
  println("El producto a*b = "+(a*b)+" no es múltiplo de N = "+N);
}
```

4.22 Dados dos números aleatorios *min* y *max*, garantiza que $min < max$.

```
//Números aleatorios entre 1 y 100
float min = random(1,100);
float max = random(1,100);
println("min = "+min+"; max = "+max);

if (min > max){
  float aux = min;
  min = max;
  max = aux;
  println("min = "+min+"; max = "+max);
} else {
  println("Ya se cumple que min < max");
}
```

4.23 Dados dos círculos determinados por sus respectivos centros y radios, determina si dichos círculos se solapan o no.

```
//Coordenadas aleatorias entre 1 y 100
float x1 = random(1,100);
float y1 = random(1,100);
float x2 = random(1,100);
float y2 = random(1,100);

//Radios aleatorios entre 10 y 30
float radio1 = random(10,30);
float radio2 = random(10,30);

println("Círculo 1 con centro en ("+x1+", "+y1+") y radio "+radio1);
println("Círculo 2 con centro en ("+x2+", "+y2+") y radio "+radio2);

//Calculamos la distancia entre ambos centros
float distanciaEntreCentros = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));

if (distanciaEntreCentros < radio1+radio2){
  println("Los círculos 1 y 2 sí se solapan");
} else {
  println("Los círculos 1 y 2 no se solapan");
}
```

4.24 Calcula el inverso de la suma de dos números, mostrando un error si se divide por cero.

```
//Coordenadas aleatorias entre 1 y 100
float x1 = random(1,100);
float y1 = random(1,100);
float x2 = random(1,100);
float y2 = random(1,100);

//Números enteros aleatorios entre -2 y 2
int x = (int)random(-2,3);
int y = (int)random(-2,3);

if (x+y != 0){
  println("1/(x+y) = 1/("+x+"+"+y+") = "+(1.0/(x+y)));
} else {
  println("No se puede dividir por cero");
}
```

4.25 Aplica un descuento del 25 % a un amigo y de un 10 % a los demás.

```
//Precio aleatorio entre 50 y 100
float precio = random(50,100);

//Descuentos
float descuentoAmigo = 0.25;
float descuentoNoAmigo = 0.1;

//Con probabilidad equiprobable se es amigo o no amigo
boolean esAmigo;
if (random(0,1) > 0.5) {
  esAmigo = true;
  println("¡Qué suerte, somos amigos!");
} else {
  esAmigo = false;
  println("Lo siento, no somos amigos");
}

float precioFinal = precio;
//Nota: fíjate que no es necesario poner esAmigo == true
if(esAmigo){
  precioFinal = precioFinal*(1-descuentoAmigo);
} else {
  precioFinal = precioFinal*(1-descuentoNoAmigo);
}
println("Del precio inicial de "+precio+" Euros, hoy vas a pagar "
    +precioFinal+" Euros, por ser tú");
```

4.26 Contar el número de divisores que tiene un número natural.

```
//Número natural aleatorio entre 50 y 100
int numero = (int)random(50,100);

int contador = 0;

//Recorremos todos los números desde 1 hasta n
for (int i = 1; i <= numero; i++){
  //Si i divide a n, entonces incrementamos el contador
  if (numero%i == 0){
```

```
    contador++;
  }
}

println("El número "+numero+" tiene "+contador+" divisores");
```

4.27 Un número abundante o número excesivo es un número x para el cual $\sigma(x) > 2x$, donde $\sigma(x)$ es la suma de todos los divisores positivos de x, incluido el propio x. Haz un programa que, dado un número natura x, indique si éste es abundante o no.

```
//Número natural aleatorio entre 1 y 100
int numero = (int)random(1,100);

int sumaDivisores = 0;

//Recorremos todos los números desde 1 hasta n
for (int i = 1; i <= numero; i++){
  //Si i divide a n, entonces lo sumamos
  if (numero%i == 0){
    sumaDivisores += i;
  }
}

if (sumaDivisores >= 2*numero){
  println("El número "+numero+" es abundante");
} else {
  println("El número "+numero+" no es abundante");
}
```

4.28 Para determinar la letra del DNI se procede como sigue:

- Se divide el número del DNI entre 23 (lo que da un resto entre 0 y 22).
- A cada resto se le asigna una letra en el siguiente orden, correspondiéndole al resto 0 la letra T y al resto 22 la letra E.

T, R, W, A, G, M, Y, F, P, D, X, B, N, J, Z, S, Q, V, H, L, C, K, E

Determina la letra de tu DNI y de 4 compañeros más.

```
//DNI aleatorio
int dni = (int)random(1,100000000);

int resto = dni%23;

char letraDni = ' ';
switch(resto){
  case 0:
    letraDni = 'T';
    break;
  case 1:
    letraDni = 'R';
    break;
  case 2:
    letraDni = 'W';
    break;
  case 3:
    letraDni = 'A';
    break;
```

```
20    case 4:
21      letraDni = 'G';
22      break;
23    case 5:
24      letraDni = 'M';
25      break;
26    case 6:
27      letraDni = 'Y';
28      break;
29    case 7:
30      letraDni = 'F';
31      break;
32    case 8:
33      letraDni = 'P';
34      break;
35    case 9:
36      letraDni = 'D';
37      break;
38    case 10:
39      letraDni = 'X';
40      break;
41    case 11:
42      letraDni = 'B';
43      break;
44    case 12:
45      letraDni = 'N';
46      break;
47    case 13:
48      letraDni = 'J';
49      break;
50    case 14:
51      letraDni = 'Z';
52      break;
53    case 15:
54      letraDni = 'S';
55      break;
56    case 16:
57      letraDni = 'Q';
58      break;
59    case 17:
60      letraDni = 'V';
61      break;
62    case 18:
63      letraDni = 'H';
64      break;
65    case 19:
66      letraDni = 'L';
67      break;
68    case 20:
69      letraDni = 'C';
70      break;
71    case 21:
72      letraDni = 'K';
73      break;
74    case 22:
75      letraDni = 'E';
```

```
76      break;
77  }

79  println("NIF = "+dni+"-"+letraDni);
```

4.29 Dados 3 valores numéricos almacenados en las variables a, b y c, ordena los valores de tal forma que al finalizar el proceso se cumpla que $a < b < c$. Imprime el mensaje correspondiente junto con los valores de a, b y c antes y después de la ordenación.

```
1   //3 números naturales aleatorios entre 1 y 10
2   int a = (int)random(1,11);
3   int b = (int)random(1,11);
4   int c = (int)random(1,11);

6   println("a = "+a+" b = "+b+" c = "+c);

8   if (a < b) {
9     if (c < a) {
10      //c<a<b
11      int aux = a;
12      a = c;
13      c = b;
14      b = aux;
15    } else if (c < b) {
16      //a<c<b
17      int aux = b;
18      b = c;
19      c = aux;
20    }
21  } else {
22    if (c < b) {
23      //c<b<a
24      int aux = c;
25      c = a;
26      a = aux;
27    } else if (a < c) {
28      //b<a<c
29      int aux = b;
30      b = a;
31      a = aux;
32    } else {
33      //b<c<a
34      int aux = b;
35      b = c;
36      c = a;
37      a = aux;
38    }
39  }

41  println("a = "+a+" b = "+b+" c = "+c);
```

4.30 Como norma general, el límite de velocidad en España para automóviles y motos, dependiendo del tipo de vía, es el siguiente:

- En vías urbanas con un único carril por sentido de circulación, 30 Km/h
- En vías urbanas con más de un carril por sentido de circulación, 50 Km/h
- En carreteras convencionales, 90 Km/h

- En autovías y autopistas, 120 Km/h

Dado un tipo de vía aleatorio y dada una velocidad aleatoria expresada en m/s, indicar si hay multa o no.

En caso de que haya multa, la base son 200 Euros, pero la cantidad final se calcula proporcionalmente al porcentaje en que se supera el límite de velocidad[1]. Por ejemplo, si la velocidad real supera en un 20 % el límite, entonces la multa será de 300 Euros (250 euros de base, más un 20 %).

```
1  //Codificamos el tipo de vía como enteros
2  final int VIA_URBANA_1_CARRIL = 0;
3  final int VIA_URBANA_MAS_CARRILES = 1;
4  final int CARRETERA_CONVENCIONAL = 2;
5  final int AUTOVIA_AUTOPISTA = 3;

7  //Vía aleatoria
8  int tipoVia = (int)random(0,4);

10 //Velocidad aleatoria entre 1 y 40 m/s
11 float velocidadMS = random(1,40);

13 //Pasamos de m/s a Km/h
14 float velocidadKMH = velocidadMS*3.6;
15 print("Circulas a "+velocidadKMH+" Km/h por ");

17 float baseMulta = 200.0;
18 float velocidadMaxima = -1;
19 switch(tipoVia){
20   case VIA_URBANA_1_CARRIL:
21     println("una vía urbana con un carril");
22     velocidadMaxima = 30.0;
23     break;
24   case VIA_URBANA_MAS_CARRILES:
25     println("una vía urbana con más de un carril");
26     velocidadMaxima = 50.0;
27     break;
28   case CARRETERA_CONVENCIONAL:
29     println("una carretera convencional");
30     velocidadMaxima = 90.0;
31     break;
32   case AUTOVIA_AUTOPISTA:
33     println("una autovía o autopista");
34     velocidadMaxima = 120.0;
35     break;
36 }

38 println("La velocidad máxima en esta vía es de "+velocidadMaxima+"
       Km/h");
39 if (velocidadKMH > velocidadMaxima){
40   float multa = baseMulta*(velocidadKMH/velocidadMaxima);
41   println("Llevas tú mucha prisa. Pasa por caja y paga una multa de "
         +multa+" Euros");
42 } else {
43   println("Circule");
```

[1] Esta forma de calcular las multas es no se corresponde con la realidad, sino que sirve únicamente para realizar el ejercicio de programación.

```
}
```

4.31 En un videojuego, un jugador gana puntos comiendo caramelos con forma circular. Los puntos que da cada caramelo dependen de su color y su área. Si el caramelo es rojo, da 200 puntos, si es verde o su área es mayor de 20, entonces recibe 150 puntos, si es amarillo y su área es menor que 20, entonces recibe 100 puntos. En cualquier otro caso, recibe 50 puntos. Haz un programa que genere un caramelo aleatorio de un determinado color ("Rojo", "Amarillo" o "Verde") con un radio aleatorio entre 2 y 3, e indique cuántos puntos ganaría el jugador si se lo comiese.

```
//Codificamos el color de los caramelos como enteros
final int ROJO = 0;
final int AMARILLO = 1;
final int VERDE = 2;

//Color aleatorio
int colorCaramelo = (int)random(0,3);
switch(colorCaramelo){
  case ROJO:
    print("Te has comido un caramelo rojo ");
    break;
  case AMARILLO:
    print("Te has comido un caramelo amarillo ");
    break;
  case VERDE:
    print("Te has comido un caramelo verde ");
    break;
}

//Radio aleatorio entre 2 y 3
float radio = random(2,3);

final float PI = 3.14;
float area = PI*radio*radio;
println("con un área de "+area);

float puntos;
if (colorCaramelo == ROJO){
  puntos = 200;
} else if ((colorCaramelo == VERDE) || (area > 20)){
  puntos = 150;
} else if ((colorCaramelo == AMARILLO) && (area < 20)){
  puntos = 100;
} else {
  puntos = 50;
}
println("La puntuación obtenida son "+puntos+" puntos");
```

4.32 El factorial de un número n se calcula como $n! = n \cdot (n-1) \cdot \ldots \cdot 1$ (por ejemplo, $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$). Haz un programa que calcule el factorial de un número natural dado n.

```
//Número entero aleatorio entre 3 y 10
int n = (int)random(3,11);

int factorial = 1;
```

```
6 for (int i = n; i > 0; i--) {
7   factorial = factorial*i;
8 }
9 println("n! = "+n+"! = "+factorial);
```

Tema 5. Programación modular (Soluciones)

5.1 Tema 5: Ejercicios para clase de teoría

5.1 Define una función que calcule la suma de dos números enteros. Haz un programa que invoque a dicha función con dos sumandos aleatorios entre 1 y 50.

```
1  int suma(int a, int b) {
2    int resultado = a + b;
3    return resultado;
4  }

6  void setup(){
7    int x = (int)random(1,50);
8    int y = (int)random(1,50);

10   int z = suma(x,y);

12   println("x + y = "+x+" + "+y+" = "+z);
13 }
```

5.2 Define una función que, dado un número x y un intervalo $[a,b]$, indique si $x \in [a,b]$ o no. Prueba tu función N veces (con $N = 10$) con parámetros aleatorios.

```
1  boolean estaContenido(int x, int min, int max) {
2    if ((x >= min) && (x <= max)){
3      return true;
4    } else {
5      return false;
6    }
7  }

9  void setup(){
10   int N = 10;

12   for (int i = 0; i < N; i++){
13     int x = (int)random(1,50);
```

```
14      int a = (int)random(1,50);
15      int b = a + (int)random(1,50);

17      if (estaContenido(x, a, b)){
18        println("x = "+x+" sí está contenido en el intervalo ["+a+", "
              +b+"]");
19      } else {
20        println("x = "+x+" no está contenido en el intervalo ["+a+", "
              +b+"]");
21      }
22    }
23  }
```

5.3 Define una función que, dado un número natural, indica si es primo o no. Prueba N (con $N = 20$) números aleatorios entre 2 y 100 y muestra un mensaje diciendo si son números primos o no.

```
1   boolean esPrimo(int n) {
2     for (int i = 2; i < n; i++) {
3       if (n%i == 0) {
4         return false;
5       }
6     }
7     return true;
8   }

10  void setup(){
11    int N = 20;

13    for (int i = 0; i < N; i++){
14      int x = (int)random(2,100);

16      if (esPrimo(x)){
17        println("El número x = "+x+" es primo");
18      } else {
19        println("El número x = "+x+" no es primo");
20      }
21    }
22  }
```

5.4 Define una función que, dado un nombre, una edad, el sexo y una dirección de correo electrónico, muestre un mensaje con toda la información asociada al perfil de esa persona. Para probar tu función, pide todos estos parámetros al usuario.

```
1   import javax.swing.JOptionPane;

3   void mostrarPerfil(String nombre, int edad, boolean sexo, String
        email) {
4     println("Nombre: "+nombre);
5     println("Edad: "+edad+" años");
6     if (sexo == true){
7       println("Sexo: Varón");
8     } else {
9       println("Sexo: Mujer");
10    }
11    println("Email: "+email);
```

```
}

void setup(){
  String nombre = JOptionPane.showInputDialog("Introduce un nombre: "
      );
  //¿Qué pasaría si aquí el usuario no introduce un número?
  int edad = int(JOptionPane.showInputDialog("Introduce una edad: "));
  //¿Qué pasaría si aquí el usuario introduce algo distinto a Varón o Mujer?
  String genero = JOptionPane.showInputDialog("Introduce un sexo
      (Varón o Mujer): ");
  boolean sexo = false;
  if (genero.equals("Varón")){
    sexo = true;
  }
  String email = JOptionPane.showInputDialog("Introduce una dirección
      de email: ");

  mostrarPerfil(nombre, edad, sexo, email);
}
```

5.5 Define una función que calcule el área de un círculo a partir de su radio. A partir de dicha función, define otra función que calcule la superficie de un cilindro dadas su altura y su radio. Prueba esta última función para un radio y una altura aleatorios.

```
float areaCirculo(float radio) {
  float area = PI*radio*radio;
  return area;
}

float superficieCilindro(float radio, float altura) {
  float areaBase = areaCirculo(radio);
  float superficie = 2*PI*radio*altura + 2*areaBase;
  return superficie;
}

void setup(){
  float radio = random(2, 10);
  float altura = random(20, 30);

  float superficieCilindro = superficieCilindro(radio, altura);

  println("La superficie del cilindro de radio "+radio+" y altura "
      +altura+" es "+superficieCilindro);
}
```

5.6 Define una función que muestre la tabla de multiplicar del número entero x, para los primeros n productos. Pide al usuario los valores de x y n.

```
import javax.swing.JOptionPane;

void tablaMultiplicar(int numero, int longitudTabla) {
  for (int i = 1; i <= longitudTabla; i++) {
    println(numero+" * "+i+" = "+(numero*i));
  }
}

void setup(){
```

```
10    JOptionPane.showMessageDialog(null, "Vamos a multiplicar el número
          x, n veces");
11    int x = int(JOptionPane.showInputDialog("Introduce el valor de x: "
          ));
12    int n = int(JOptionPane.showInputDialog("Introduce el valor de n: "
          ));

14    tablaMultiplicar(x, n);
15  }
```

5.7 Define una función que indique si un número dado n es múltiplo o no de otro dado m. Haz un programa que pida al usuario el valor de m y que luego genere números aleatorios para n en el rango $[0, 20 \cdot m]$ hasta encontrar N (con $N = 10$) múltiplos de m.

```
1   import javax.swing.JOptionPane;

3   boolean esMultiplo(int n, int m) {
4     if ((n > 0) && (n%m == 0)){
5       return true;
6     }
7     return false;
8   }

10  void setup(){
11    int N = 10;
12    int m = int(JOptionPane.showInputDialog("Introduce el valor de m: "
          ));
13    JOptionPane.showMessageDialog(null, "Vamos a buscar "+N+" múltiplos
          de "+m+" en el intevalo [0, "+(20*m)+"]");
14    int numMultiplos = 0;

16    while (numMultiplos < N) {
17      int n = (int)random(0, 20*m);

19      if (esMultiplo(n, m)) {
20        println("Bingo! El "+n+" es múltiplo de "+m);
21        numMultiplos++;
22      }
23    }
24  }
```

5.8 Define una función que, dado un entero n, indique el número de cifras que lo componen. Prueba tu función con N (para $N = 20$) números aleatorios entre 1 y 10^4.

```
1   int numCifras(int numero) {
2     int aux = numero;
3     int cifras = 1;

5     while(aux > 10){
6       cifras++;
7       aux = aux/10;
8     }
9     return cifras;
10  }

12  void setup(){
13    int N = 20;
```

```
  for (int i = 0; i < N; i++){
    int n = (int)random(1, 10000);
    int cifras = numCifras(n);
    println("El número "+n+" tiene "+cifras+" cifras");
  }
}
```

5.9 Dada una variable `s` de tipo `String`, es posible conocer la longitud de la cadena de la siguiente forma:

```
String s = "Cadena";
int longitud = s.length(); //La variable longitud almacenará un 6
```

Por otra parte, es posible conocer el carácter que ocupa la posición *i*-ésima, de la siguiente forma:

```
String s = "Cadena";
char caracter = s.charAt(0); //La variable caracter almacenará una 'C'
caracter = s.charAt(4); //La variable caracter almacenará ahora una 'n'
```

Sabiendo esto, haz una función que, dada una cadena de caracteres, calcule y devuelva el número de vocales que contiene. Prueba tu función con varias cadenas de caracteres que representen un refrán a tu elección.

```
import javax.swing.JOptionPane;

int numVocales(String cadena) {
  int vocales = 0;

  for (int i = 0; i < cadena.length(); i++){
    char caracter = cadena.charAt(i);
    if (caracter == 'a' || caracter == 'A' ||
    caracter == 'á' || caracter == 'Á' ||
    caracter == 'e' || caracter == 'E' ||
    caracter == 'é' || caracter == 'É' ||
    caracter == 'i' || caracter == 'I' ||
    caracter == 'í' || caracter == 'Í' ||
    caracter == 'o' || caracter == 'O' ||
    caracter == 'ó' || caracter == 'Ó' ||
    caracter == 'u' || caracter == 'U' ||
    caracter == 'ú' || caracter == 'Ú' ||
    caracter == 'ü'){
      vocales++;
    }
  }
  return vocales;
}

void setup(){
  String refran1 = "No por antes madrugar, amanece más temprano";
  String refran2 = "Cuando veas a tu vecino las barbas quemar, pon
      las tuyas a remojar";
  String refran3 = "A quien madruga, Dios le ayuda";

  println("El refrán \""+refran1+"\" tiene "+numVocales(refran1)+"
      vocales");
```

```
31    println("El refrán \""+refran2+"\" tiene "+numVocales(refran2)+"
          vocales");
32    println("El refrán \""+refran3+"\" tiene "+numVocales(refran3)+"
          vocales");
33  }
```

5.2 Tema 5: Ejercicios para clase de prácticas

5.10 Define una función que, usando la función `random(0,1)` (que devuelve un número aleatorio en el intervalo $[0,1)$), devuelva un número aleatorio entre dos enteros a y b dados. Prueba a ejecutar tu función N veces (con $N = 20$) para $a = 30$ y $b = 50$.

```
int aleatorio(int min, int max) {
  int numeroAleatorio = min + (int)(random(0,1.0001)*(max-min));
  return numeroAleatorio;
}

void setup(){
  int N = 20;
  int a = 30;
  int b = 50;

  println("Vamos a buscar "+N+" números aleatorios en el intervalo
      ["+a+", "+b+"]");
  println("Por ejemplo...");
  for (int i = 0; i < N; i++) {
    int x = aleatorio(a, b);
    println(x);
  }
}
```

5.11 Define una función que calcule la distancia euclídea entre dos puntos dados (x_1, y_1), (x_2, y_2). Prueba dicha función generando dos puntos aleatorios con la función del ejercicio **5.10**.

```
int aleatorio(int min, int max) {
  int numeroAleatorio = min + (int)(random(0,1.0001)*(max-min));
  return numeroAleatorio;
}

float distanciaEuclidea(int x1, int y1, int x2, int y2) {
  float distancia = sqrt(pow(x1-x2, 2) + pow(y1-y2, 2));
  return distancia;
}

void setup(){
  int x1 = aleatorio(-100, 100);
  int x2 = aleatorio(-100, 100);
  int y1 = aleatorio(-100, 100);
  int y2 = aleatorio(-100, 100);

  println("La distancia entre los puntos ("+x1+", "+y1+") y ("+x2+", "
      +y2+") = "+distanciaEuclidea(x1, y1, x2, y2));
}
```

5.12 Define una función que, dados dos círculos (definidos por sus respectivos centros y radios), indique si dichos círculos se solapan o no. Prueba tu función con círculos aleatorios cuyas coordenadas de sus centros tomen valores en el intervalo $[-100, 100]$ y cuyos radios tomen valores en el intervalo $[1, 10]$ hasta encontrar N (con $N = 25$) círculos que sí se solapen.

```
float distanciaEuclidea(int x1, int y1, int x2, int y2) {
  float distancia = sqrt(pow(x1-x2, 2) + pow(y1-y2, 2));
  return distancia;
}
```

```
boolean circulosSolapan(int x1, int y1, float radio1, int x2, int y2,
    float radio2){
  if (distanciaEuclidea(x1, y1, x2, y2) < radio1+radio2) {
    return true;
  }
  return false;
}

void setup(){
  int N = 25;
  int numCirculosSolapan = 0;

  while (numCirculosSolapan < N) {
    int x1 = (int)random(-100, 100);
    int x2 = (int)random(-100, 100);
    float radio1 = random(1,10);

    int y1 = (int)random(-100, 100);
    int y2 = (int)random(-100, 100);
    float radio2 = random(1,10);

    if (circulosSolapan(x1, y1, radio1, x2, y2, radio2)) {
      println("El círculo con centro en ("+x1+", "+y1+") y radio "
          +radio1+" y el círculo con centro en ("+x2+", "+y2+") y
          radio "+radio2+" se solapan");
      numCirculosSolapan++;
    }
  }
}
```

5.13 Define una función que muestra por pantalla los divisores de un número natural n dado. Prueba tu función N veces (con $N = 15$) para un valor n aleatorio entre 2 y 200.

```
void mostrarDivisores(int numero) {
  print("Los divisores del "+numero+" son: ");
  for (int i = 1; i < numero; i++){
    if (numero%i == 0) {
      print(i+", ");
    }
  }
  println(numero);
}

void setup(){
  int N = 15;

  for (int i = 0; i < N; i++){
    int n = (int)random(2, 200);
    mostrarDivisores(n);
  }
}
```

5.14 Dos números amigos son dos números enteros positivos a y b tales que la suma de los divisores propios de uno es igual al otro número y viceversa. Define una función que determine si dos números son amigos o no. A partir de la función anterior, haz las siguientes funciones:

- Una función que busque aleatoriamente N (con $N = 5$) pares de números amigos en el rango $[A,B] = [1,500]$.
- Una función que busque sistemáticamente todos los pares de números amigos en el rango $[A,B] = [1,500]$.

```
1  int sumaDivisoresPropios(int numero) {
2    int suma = 0;
3    for (int i = 1; i < numero; i++){
4      if (numero%i == 0) {
5        suma += i;
6      }
7    }
8    return suma;
9  }

11 boolean sonNumerosAmigos(int a, int b){
12   return ((sumaDivisoresPropios(a) == b)
13   && (sumaDivisoresPropios(b) == a));
14 }

16 void buscarNumerosAmigos(int cantidadAmigos, int min, int max) {
17   int amigosEncontrados = 0;

19   while(amigosEncontrados < cantidadAmigos) {
20     int numero1 = (int)random(min, max);
21     int numero2 = (int)random(min, max);

23     if((numero1 != numero2) && sonNumerosAmigos(numero1, numero2)) {
24       println("Los números "+numero1+" y "+numero2+" son amigos");
25       amigosEncontrados++;
26     }
27   }
28 }

30 void buscarTodosNumerosAmigos(int min, int max) {
31   println("Todos los números amigos en el intervalo ["+min+", "
         +max+"] son: ");
32   for (int i = min; i <= max; i++) {
33     for (int j = i; j <= max; j++) {
34       if((i != j) && sonNumerosAmigos(i,j)) {
35         println("("+i+", "+j+")");
36       }
37     }
38   }
39 }

41 void setup(){
42   int N = 5;
43   int min = 1;
44   int max = 500;

46   buscarNumerosAmigos(N, min, max);
47   println();
48   buscarTodosNumerosAmigos(min, max);
49 }
```

5.15 Haz una función que, dado un número natural *n*, devuelva otro número que sea igual a la suma de las cifras de *n*.

```
int sumaCifras(int numero) {
  int aux = numero;
  int suma = 0;

  while(aux >= 10){
    suma += aux%10;
    aux = aux/10;
  }
  suma += aux;
  return suma;
}

void setup(){
  int N = 20;

  for (int i = 0; i < N; i++){
    int n = (int)random(1, 10000);
    int suma = sumaCifras(n);
    println("La suma de las cifras del número "+n+" es "+suma);
  }
}
```

5.16 Un palíndromo[1] es una palabra o frase que se lee igual en un sentido que en otro (por ejemplo, "Yo hago yoga hoy"). Haz una función que, dada una cadena de caracteres, indique si se trata de un palíndromo o no.

```
boolean esPalindromo(String cadena) {
  int desplazamiento = 0;
  for (int i = 0; i <= cadena.length()/2; i++){
    int pos1 = i;
    int pos2 = cadena.length()-1-i+desplazamiento;

    while((cadena.charAt(pos1) == ' ') && (pos1 < cadena.length())){
      pos1++;
      i++;
      desplazamiento++;
    }

    while((cadena.charAt(pos2) == ' ') && (pos2 >= 0)){
      pos2--;
      desplazamiento--;
    }

    if (cadena.charAt(pos1) != cadena.charAt(pos2)) {
      return false;
    }
  }
  return true;
}

void setup(){
  String cadena1 = "yo hago yoga hoy";
```

[1] https://es.wikipedia.org/wiki/PalÃŋndromo

```
27    String cadena2 = "Yo no hago yoga ni hoy, ni nunca";

29    if (esPalindromo(cadena1)) {
30      println("La cadena \""+cadena1+"\" es un palíndromo");
31    } else {
32      println("La cadena \""+cadena1+"\" no es un palíndromo");
33    }

35    if (esPalindromo(cadena2)) {
36      println("La cadena \""+cadena2+"\" es un palíndromo");
37    } else {
38      println("La cadena \""+cadena2+"\" no es un palíndromo");
39    }
40  }
```

5.17 Haz una función que construya y devuelva una cadena de caracteres de longitud dada *n*. Cada carácter de la cadena será, o bien un paréntesis abierto '(', o bien un paréntesis cerrado ')', con probabilidad equiprobable.

```
1  String parentesis(int longitud) {
2    String cadena = "";

4    for (int i = 0; i < longitud; i++){
5      if (random(0,1) > 0.5){
6        cadena += "(";
7      } else {
8        cadena += ")";
9      }
10   }
11   return cadena;
12 }

14 void setup(){
15   int N = 10;

17   for (int i = 0; i < N; i++) {
18     int n = (int)random(5,10);
19     String cadena = parentesis(n);
20     println(cadena);
21   }
22 }
```

5.3 Tema 5: Ejercicios para casa

5.18 Haz una función que, dados tres puntos (x_i, y_i) (con $i \in \{1,2,3\}$), calcule y devuelva el área del triángulo que conforman. En caso de que esos tres puntos no conformen un triángulo, según el Teorema de la desigualdad, la función devolverá -1 como "código de error".

```
/*
 * Función para calcular la distancia Euclídea entre
 * dos puntos (x1, y1) y (x2, y2)
 */
float distanciaEuclidea(int x1, int y1, int x2, int y2) {
  float distancia = sqrt(pow(x1-x2, 2) + pow(y1-y2, 2));
  return distancia;
}

/*
 * Esta función comprueba si los tres puntos introducidos
 * como parámetros forman o no un triángulo aplicando el
 * Teorema de la desigualdad triangular
 */
boolean formanTriangulo(int x1, int y1, int x2, int y2, int x3, int
    y3) {
  float lado1 = distanciaEuclidea(x1, y1, x2, y2);
  float lado2 = distanciaEuclidea(x2, y2, x3, y3);
  float lado3 = distanciaEuclidea(x1, y1, x3, y3);

  return ((lado1 < (lado2 + lado3)) &&
          (lado2 < (lado1 + lado3)) &&
          (lado3 < (lado1 + lado2)));
}

/*
 * Calculamos el área de un triángulo usando la Fórmula
 * de Herón.
 */
float areaTriangulo(int x1, int y1, int x2, int y2, int x3, int y3) {
  //Si los tres puntos dados no forman un triángulo, devolvemos -1
  if (!formanTriangulo(x1, y1, x2, y2, x3, y3)) {
    return -1;
  }

  float lado1 = distanciaEuclidea(x1, y1, x2, y2);
  float lado2 = distanciaEuclidea(x2, y2, x3, y3);
  float lado3 = distanciaEuclidea(x1, y1, x3, y3);

  float semiperimetro = (lado1+lado2+lado3)/2.0;
  return sqrt(semiperimetro *
              (semiperimetro - lado1) *
              (semiperimetro - lado2) *
              (semiperimetro - lado3));
}

void setup(){
  int x1 = (int)random(-100, 100);
  int x2 = (int)random(-100, 100);
  int x3 = (int)random(-100, 100);
  int y1 = (int)random(-100, 100);
```

```
  int y2 = (int)random(-100, 100);
  int y3 = (int)random(-100, 100);

  float area = areaTriangulo(x1, y1, x2, y2, x3, y3);
  if (area != -1) {
    println("Los puntos ("+x1+", "+y1+"), ("+x2+", "+y2+") y ("+x3+",
        "+y3+") forman un triángulo y su área es "+area);
  } else {
    println("Los puntos ("+x1+", "+y1+"), ("+x2+", "+y2+") y ("+x3+",
        "+y3+") no forman un triángulo ");
  }
}
```

5.19 Haz una función que, dado un número natural n, devuelva otro número que sea igual a la suma de las cifras de n que ocupan una posición par menos la suma de las cifras que ocupan una posición impar (las unidades ocuparían la posición 1, las decenas la posición 2, etc.). Por ejemplo, para el caso de 28.075.413, tendríamos que:

$$28{,}075{,}413 \rightarrow -14 = (1+5+0+2) - (3+4+7+8)$$

```
int sumaLocaCifras(int numero) {
  int suma = 0;
  int coeficiente = -1;

  int aux = numero;
  while (aux > 0){
    suma += coeficiente * (aux%10);
    aux = aux/10;
    coeficiente = -coeficiente;
  }

  return suma;
}

void setup() {
  int N = 10;
  int min = 1000;
  int max = 100000;

  for (int i = 0; i < N; i++) {
    int numero = (int)random(min, max);
    int sumaLoca = sumaLocaCifras(numero);
    println("La suma loca de las cifras de "+numero+" es "+sumaLoca);
  }
}
```

5.20 El último Teorema de Fermat dice que si n es un número entero mayor o igual que 3, entonces no existen números enteros positivos x, y y z, tales que se cumpla la igualdad:

$$x^n + y^n = z^n$$

Haz una función que dados dos números naturales n y max, compruebe si se cumple o no el Teorema de Fermat $\forall x, y, z \in [1, max]$. Prueba la función con $n \in \{3,4\}$ y $max \in \{25, 50, 100\}$.

```
boolean igualdadFermat(int x, int y, int z, int n){
  return ((int)pow(x, n) + (int)pow(y, n) == (int)pow(z, n));
}

boolean teoremaFermat(int n, int max){
  for (int x = 1; x <= max; x++) {
    for (int y = 1; y <= max; y++) {
      for (int z = 1; z <= max; z++) {
          if (igualdadFermat(x, y, z, n)) {
            return false;
          }
      }
    }
  }
  return true;
}

void setup() {

  for (int n = 3; n <= 4; n++) {
    for (int max = 25; max <= 100; max = max*2) {
      if (teoremaFermat(n, max)) {
        println("El Teorema de Fermat se cumple para n = "+n+" con x,
            y, z entre 1 y "+max);
      } else {
        println("El Teorema de Fermat no se cumple para n = "+n+" con
            x, y, z entre 1 y "+max);
        println("Por favor, pasa a recoger tu Medalla Fields");
      }
    }
  }
}
```

5.21 Haz una función que, dado un número natural n, calcule su factorial $n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$. Prueba tu función con los primeros 6 números naturales.

```
int factorial(int numero) {
  int factorial = 1;

  for (int i = numero; i > 0; i--) {
     factorial = factorial * i;
  }

  return factorial;
}

void setup() {
  for (int n = 1; n < 7; n++) {
    println("n! = "+n+"! = "+factorial(n));
  }
}
```

5.22 El coeficiente binomial[2] $\binom{n}{k}$ se calcula como

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Haz una función que, a partir de la función del ejercicio **5.21**, calcule el coeficiente binomial $\binom{n}{k}$.

```
int factorial(int numero) {
  int factorial = 1;

  for (int i = numero; i > 0; i--) {
     factorial = factorial * i;
  }

  return factorial;
}

/*
 * Esta función calcula el coeficiente binomial
 * n sobre k
 */
int coeficienteBinomial(int n, int k){
  if (n == 0) {
    return 0;
  }
  return factorial(n)/(factorial(k)*factorial(n-k));
}

void setup() {
  for (int n = 0; n < 10; n++) {
    for (int k = 0; k < n; k++) {
      println("C(n, k) = C("+n+", "+k+") = "+coeficienteBinomial(n,
          k));
    }
  }
}
```

5.23 El Teorema del binomio[3] establece que cualquier potencia de un binomio $x+y$ puede ser expandida en una suma de la forma:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

Sabiendo esto, haz una función que, haciendo uso de la función del ejercicio **5.22**, y dados los números reales x, y y el número natural n, calcule la potencia $(x+y)^n$. Prueba tu función con valores aleatorios de $x, y \in [1,10]$ y $n \in \{2,3,4,5\}$.

```
int factorial(int numero) {
  int factorial = 1;

  for (int i = numero; i > 0; i--) {
```

[2] https://es.wikipedia.org/wiki/Coeficiente_binomial
[3] https://es.wikipedia.org/wiki/Teorema_del_binomio

```
5       factorial = factorial * i;
6     }

8     return factorial;
9   }

11  /*
12   * Esta función calcula el coeficiente binomial
13   * n sobre k
14   */
15  int coeficienteBinomial(int n, int k){
16    if (n == 0) {
17      return 0;
18    }
19    return factorial(n)/(factorial(k)*factorial(n-k));
20  }

22  /*
23   * Esta función calcula la n-ésima potencia del
24   * binomio (x+y) usando el Teorema del Binomio
25   */
26  int teoremaBinomio(int x, int y, int n) {
27    int resultado = 0;

29    for (int k = 0; k <= n; k++) {
30      resultado += coeficienteBinomial(n, k) *
31                   pow(x, n-k) *
32                   pow(y, k);
33    }

35    return resultado;
36  }

38  void setup() {
39    for (int n = 2; n < 6; n++) {
40      int x = (int)random(1,11);
41      int y = (int)random(1,11);
42      println("(x + y)^n = ("+x+" + "+y+")^"+n+" = "+teoremaBinomio(x,
            y, n));
43    }
44  }
```

5.24 El número áureo (también llamado número de oro, número de Dios, razón extrema y media, razón áurea, razón dorada, media áurea, proporción áurea y divina proporción) es un número irracional, representado por la letra griega ϕ (phi) (en minúscula) o Φ (Phi) (en mayúscula) en honor al escultor griego Fidias.
Su valor numérico es:

$$\phi = \frac{1+\sqrt{5}}{2}$$

Por otra parte, se dice que dos números naturales a y b están en proporción áurea si se cumple que:

$$\frac{a+b}{a} = \frac{a}{b}$$

Haz una función que, dados dos números naturales a y b, determine si están en proporción

áurea o no. A partir de esta función, haz otra función que busque todos los pares de números a y b que están en proporción áurea dentro del intervalo $[min, max]$.

```
boolean numerosEnProporcionAurea(int a, int b) {
  return ((a+b)/(float)a == a/(float)b);
}

void buscaNumerosEnProporcionAurea(int min, int max) {
  for (int i = min; i <= max; i++) {
    for (int j = min; j <= max; j++) {
      if (numerosEnProporcionAurea(i, j)) {
         println("Los números "+i+" y "+j+" están en proporción á
             urea");
      }
    }
  }
}

void setup() {
  int min = 1;
  int max = 10000;

  buscaNumerosEnProporcionAurea(min, max);
}
```

5.25 Dada una cadena aleatoria de paréntesis abiertos y cerrados generada con la función desarrollada en el ejercicio **5.17**, haz una función que cuente cuántos paréntesis están correctamente cerrados. Por ejemplo, la cadena ‘‘)(()()(’’ tiene 2 paréntesis correctamente cerrados, mientras que la cadena ‘‘(()(()())()()’’ tiene 6.

```
String parentesis(int longitud) {
  String cadena = "";

  for (int i = 0; i < longitud; i++){
    if (random(0,1) > 0.5){
      cadena += "(";
    } else {
      cadena += ")";
    }
  }
  return cadena;
}

int contarParentesisCerrados(String parentesis) {
  int contador = 0;
  int parentesisAbiertos = 0;

  for (int i = 0; i < parentesis.length(); i++) {
    if (parentesis.charAt(i) == '(') {
      parentesisAbiertos++;
    } else if (parentesisAbiertos > 0) {
      contador++;
      parentesisAbiertos--;
    }
  }
```

```
27    return contador;
28  }

30  void setup(){
31    int N = 10;

33    for (int i = 0; i < N; i++) {
34      int n = (int)random(10,20);
35      String cadena = parentesis(n);
36      println("La cadena de paréntesis \""+cadena+
37      "\" tiene "+contarParentesisCerrados(cadena)+
38      " paréntesis correctamente cerrados");
39    }
40  }
```

Tema 6. Datos estructurados (Soluciones)

6.1 Tema 6: Ejercicios para clase de teoría

6.1 Sobre una lista de naturales (enteros positivos) construir las siguientes funciones:

- Crear una lista de longitud n con valores aleatorios entre un mínimo y un máximo.
- Crear una lista aleatoria de longitud n con los n primeros (y distintos) números naturales.
- Crear una copia de la lista.
- Limpiar todas las posiciones de la lista, asignándoles un valor dado.
- Indicar si un valor dado está contenido en la lista o no.
- Modificar los valores pares de una lista para que valgan la mitad.
- Modificar cada término según la expresión $a_i = a_{i-1} + a_i + a_{i+1}$.
- Mostrar los valores de una lista de naturales.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }
9
10   int[] lista = new int[longitud];
11
12   for (int i = 0; i < longitud; i++) {
13      lista[i] = (int)random(min, max+1);
14   }
15
16   return lista;
17 }
18
19 /*
20  * Crea una lista aleatoria de longitud n con los
21  * n primeros (y distintos) números naturales
```

```
22   */
23  int[] crearListaPrimerosNaturales(int longitud) {
24    if (longitud < 1) {
25       return null;
26    }

28    int[] lista = new int[longitud];
29    int cursor = 0;
30    while (cursor < longitud){
31       int numero = (int)random(1, longitud+1);
32       boolean repetido = false;
33       for (int i = 0; i < cursor; i++) {
34           if (lista[i] == numero) {
35             repetido = true;
36             break;
37           }
38       }
39       if (!repetido) {
40         lista[cursor] = numero;
41         cursor++;
42       }
43    }

45    return lista;
46  }

48  /*
49   * Crea una copia de una lista dada
50   */
51  int[] copiarLista(int[] lista) {
52    if (lista == null){
53      return null;
54    }

56    int longitud = lista.length;
57    int[] copia = new int[longitud];

59    for (int i = 0; i < longitud; i++) {
60       copia[i] = lista[i];
61    }

63    return copia;
64  }

66  /*
67   * Limpia una lista dada, asignando un valor
68   * dado en todas sus posiciones
69   */
70  void limpiarLista(int[] lista, int valor){
71     if ((lista != null) && (valor > 0)){
72       for (int i = 0; i < lista.length; i++){
73          lista[i] = valor;
74       }
75     }
76  }
```

```
78 /*
79  * Busca un valor dado en una lista dada.
80  * Devuelve true si encuentra el valor y
81  * false si no lo encuentra
82  */
83 boolean buscarValor(int[] lista, int valor){
84    if (lista != null) {
85      for (int i = 0; i < lista.length; i++){
86         if (lista[i] == valor) {
87            return true;
88         }
89      }
90    }
91    return false;
92 }

94 /*
95  * Dada una lista, divide por la mitad los
96  * elementos en las posiciones pares
97  */
98 void valoresParesAMitad(int[] lista){
99    if (lista != null) {
100     for (int i = 0; i < lista.length; i++){
101        if (i%2 == 0) {
102           lista[i] = lista[i]/2;
103        }
104     }
105   }
106 }

108 /*
109  * Modifica una lista dada para sumar a cada
110  * elemento su predecesor y su sucesor (considerando
111  * una lista circular)
112  */
113 void sumaPredecesorYSucesor(int[] lista){
114   if (lista != null) {
115     int[] copia = copiarLista(lista);
116     lista[0] += copia[copia.length-1] + copia[1];
117      for (int i = 1; i < lista.length; i++){
118        lista[i] += copia[i-1] + copia[(i+1)%copia.length];
119      }
120   }
121 }

123 /*
124  * Muestra por pantalla los valores de una
125  * lista dada
126  */
127 void mostrarLista(int[] lista){
128    if (lista != null) {
129      for (int i = 0; i < lista.length; i++){
130        print(lista[i]+" ");
131      }
132    }
133    println();
```

```
134 }

136 void setup(){
137   int longitud = 10;
138   int min = 5;
139   int max = 15;
140   int nuevoValor = 10;

142   int[] miLista = crearListaAleatoria(longitud, min, max);
143   println("Mi lista aleatoria de longitud "+longitud+" y valores
          entre "+min+" y "+max+" es:");
144   mostrarLista(miLista);
145   println();

147   println("Mi lista aleatoria de con los primeros "+longitud+"
          números naturales es:");
148   miLista = crearListaPrimerosNaturales(longitud);
149   mostrarLista(miLista);
150   println();

152   int[] miCopia = copiarLista(miLista);
153   println("La copia de mi úlima lista es:");
154   mostrarLista(miCopia);
155   println();

157   limpiarLista(miCopia, nuevoValor);
158   println("La copia de mi úlima lista después de limpiarla con el
          valor "+nuevoValor+" es:");
159   mostrarLista(miCopia);
160   println();

162   nuevoValor += (int)random(0,2);
163   if(buscarValor(miCopia, nuevoValor)) {
164     println("La copia limpiada de la lista original sí contiene el "
          +nuevoValor+"\n");
165   } else {
166     println("La copia limpiada de la lista original no contiene el "
          +nuevoValor+"\n");
167   }

169   valoresParesAMitad(miCopia);
170   println("Si dividimos por 2 las posiciones pares, entonces la copia
          de mi úlima lista es:");
171   mostrarLista(miCopia);
172   println();

174   sumaPredecesorYSucesor(miCopia);
175   println("Y si sumamos predecesor y sucesor, entonces la copia de mi
          úlima lista es:");
176   mostrarLista(miCopia);
177 }
```

6.2 Construye cada una de las funciones cuyo propósito se indica a continuación:

- Generar matrices aleatorias para rangos dados.
- Calcular la suma de dos matrices, si es posible.
- Modificar los valores de una matriz multiplicando cada elemento por una constante dada.

- Indicar si dos matrices son iguales o no.
- Generar y devolver una representación de tipo String de la matriz
- Mostrar los valores de una matriz.

```
1  /*
2   * Crea una matriz de un rango dado con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
       max) {
6    if ((filas < 1) || (columnas < 1) || (max < min)) {
7      return null;
8    }
9
10   int[][] matriz = new int[filas][columnas];
11
12   for (int i = 0; i < filas; i++) {
13     for (int j = 0; j < columnas; j++) {
14       matriz[i][j] = (int)random(min, max+1);
15     }
16   }
17
18   return matriz;
19 }
20
21 /*
22  * Suma dos matrices dadas, si es posible.
23  * Si la suma no es posible, devuelve null.
24  */
25 int[][] sumarMatrices(int[][] matriz1, int[][] matriz2) {
26   if ((matriz1 == null) || (matriz2 == null) ||
27       (matriz1.length != matriz2.length) ||
28       (matriz1[0].length != matriz2[0].length)) {
29     return null;
30   }
31
32   int[][] matrizSuma = new int[matriz1.length][matriz1[0].length];
33   for (int i = 0; i < matriz1.length; i++) {
34     for (int j = 0; j < matriz1[0].length; j++) {
35       matrizSuma[i][j] = matriz1[i][j] + matriz2[i][j];
36     }
37   }
38
39   return matrizSuma;
40 }
41
42 /*
43  * Modifica los valores de una matriz multiplicando
44  * cada elemento por una constante dada.
45  */
46 void productoConstanteMatriz(int[][] matriz, int constante) {
47   if (matriz == null){
48     return;
49   }
50
51   for (int i = 0; i < matriz.length; i++) {
```

```
52      for (int j = 0; j < matriz[0].length; j++) {
53        matriz[i][j] = matriz[i][j] * constante;
54      }
55    }
56  }

58  /*
59   * Comprueba si dos matrices dadas son iguales
60   * o no. Si son iguales, devuelve true, y si no
61   * son iguales, devuelve false.
62   */
63  boolean matricesIguales(int[][] matriz1, int[][] matriz2){
64    if ((matriz1 == null) || (matriz2 == null) ||
65        (matriz1.length != matriz2.length) ||
66        (matriz1[0].length != matriz2[0].length)) {
67       return false;
68    }

70    for (int i = 0; i < matriz1.length; i++) {
71      for (int j = 0; j < matriz1[0].length; j++) {
72        if (matriz1[i][j] != matriz2[i][j]) {
73          return false;
74        }
75      }
76    }

78    return true;
79  }

81  /*
82   * Genera y devuelve una representación de tipo
83   * String de una matriz dada.
84   */
85  String matrizAString(int[][] matriz){
86    String cadena = "";

88    if (matriz == null){
89      return cadena;
90    }

92    for (int i = 0; i < matriz.length; i++) {
93      for (int j = 0; j < matriz[0].length; j++) {
94        cadena += matriz[i][j] + "\t";
95      }
96      cadena += "\n";
97    }
98    return cadena;
99  }

101 /*
102  * Muestra por pantalla los valores de una
103  * matriz dada
104  */
105 void mostrarMatriz(int[][] matriz){
106    print(matrizAString(matriz));
107 }
```

```
109 void setup(){
110   int filas = 4;
111   int columnas = 5;
112   int min = 5;
113   int max = 15;
114   int constante = 10;

116   int[][] miMatriz1 = crearMatrizAleatoria(filas, columnas, min, max);
117   println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
          columnas, con valores entre "+min+" y "+max+" es:");
118   mostrarMatriz(miMatriz1);
119   println();

121   int[][] miMatriz2 = crearMatrizAleatoria(filas, columnas, min, max);
122   println("Mi matriz aleatoria B con "+filas+" filas y "+columnas+"
          columnas, con valores entre "+min+" y "+max+" es:");
123   mostrarMatriz(miMatriz2);
124   println();

126   int[][] miMatrizSuma = sumarMatrices(miMatriz1, miMatriz2);
127   println("La suma de las matrices A y B es:");
128   mostrarMatriz(miMatrizSuma);
129   println();

131   productoConstanteMatriz(miMatrizSuma, constante);
132   println("El resulado de multiplicar la matriz suma por la constante
          "+constante+" es:");
133   mostrarMatriz(miMatrizSuma);
134   println();

136   if(matricesIguales(miMatriz1, miMatriz2)) {
137      println("Las matrices A y B sí son iguales\n");
138   } else {
139      println("Las matrices A y B no son iguales\n");
140   }
141 }
```

6.3 Crea un registro para representar a un Alumno con, al menos los siguientes campos: nombre, apellidos, fecha de nacimiento, sexo, email.
Haz un programa que represente 3 alumnos y muestra su información por pantalla.

6.4 Crea un registro para representar una asignatura con, al menos, los siguientes campos: nombre, créditos, cuatrimestre y calificación (de 0 a 10).
Amplía el ejercicio **6.3** para que cualquier alumno tenga 3 asignaturas.

6.2 Tema 6: Ejercicios para clase de prácticas

6.5 Dado un array de enteros, implementar las siguientes funciones:

- Calcular la media aritmética de todos sus elementos
- Calcular la desviación típica.

```
/*
 * Crea una lista de una longitud dada con valores
 * aleatorios entre un mínimo y un máximo
 */
int[] crearListaAleatoria(int longitud, int min, int max) {
  if ((longitud < 1) || (max < min)) {
    return null;
  }

  int[] lista = new int[longitud];

  for (int i = 0; i < longitud; i++) {
     lista[i] = (int)random(min, max+1);
  }

  return lista;
}

/*
 * Muestra por pantalla los valores de una
 * lista dada
 */
void mostrarLista(int[] lista){
   if (lista != null) {
     for (int i = 0; i < lista.length; i++){
       print(lista[i]+" ");
     }
   }
   println();
}

/*
 * Calcula la media aritmética de los elementos
 * de una lista dada
 */
float mediaAritmeticaLista(int[] lista) {
   float media = 0;

   if ((lista == null) || (lista.length < 1)) {
     return media;
   }

  for (int i = 0; i < lista.length; i++) {
     media += lista[i];
  }

  media = media/(float)lista.length;
  return media;
}

/*
```

```
52  * Calcula la desviación típica de los elementos
53  * de una lista dada
54  */
55 float desviacionTipicaLista(int[] lista) {
56    float desviacion = 0;

58    if ((lista == null) || (lista.length < 1)) {
59      return desviacion;
60    }

62    float media = mediaAritmeticaLista(lista);
63    for (int i = 0; i < lista.length; i++) {
64      desviacion += pow(lista[i] - media, 2);
65    }

67   desviacion = sqrt(desviacion/lista.length);
68   return desviacion;
69 }

71 void setup(){
72   int longitud = 10;
73   int min = 5;
74   int max = 15;

76   int[] miLista = crearListaAleatoria(longitud, min, max);
77   println("Mi lista aleatoria de longitud "+longitud+" y valores
         entre "+min+" y "+max+" es:");
78   mostrarLista(miLista);
79   println();

81   println("La media aritmética de mi lista aleatoria es: "
         +mediaAritmeticaLista(miLista)+"\n");

83   println("Y la desviación típica de mi lista aleatoria es: "
         +desviacionTipicaLista(miLista)+"\n");
84 }
```

6.6 Dados dos vectores $\overrightarrow{u}$ y $\overrightarrow{v}$, implementar las siguientes funciones:

- Calcular el módulo de un vector
- Sumar ambos vectores
- Restar ambos vectores
- Multiplicar un vector por un escalar
- Producto escalar de dos vectores

```
1 /*
2  * Crea un vector de una longitud dada con valores
3  * aleatorios entre un mínimo y un máximo
4  */
5 int[] crearVectorAleatorio(int longitud, int min, int max) {
6   if ((longitud < 1) || (max < min)) {
7     return null;
8   }

10  int[] vector = new int[longitud];

12  for (int i = 0; i < longitud; i++) {
```

```
13      vector[i] = (int)random(min, max+1);
14    }

16    return vector;
17  }

19  /*
20   * Muestra por pantalla los valores de un
21   * vector dado
22   */
23  void mostrarVector(int[] vector){
24    if (vector != null) {
25      print("(");
26      if (vector.length > 0) {
27        for (int i = 0; i < vector.length-1; i++){
28          print(vector[i]+", ");
29        }
30        print(vector[vector.length-1]);
31      }
32      print(")");
33    }
34    println();
35  }

37  /*
38   * Calcula el módulo de un vector dado
39   */
40  float moduloVector(int[] vector) {
41    float modulo = 0;

43    if ((vector == null) || (vector.length < 1)) {
44      return modulo;
45    }

47    for (int i = 0; i < vector.length; i++) {
48      modulo += pow(vector[i], 2);
49    }

51    modulo = sqrt(modulo);
52    return modulo;
53  }

55  /*
56   * Calcula la suma de dos vectores dados
57   */
58  int[] sumaVectores(int[] vector1, int[] vector2) {
59    if ((vector1 == null) || (vector2 == null) ||
60          (vector1.length != vector2.length)) {
61      return null;
62    }

64    int[] vectorSuma = new int[vector1.length];
65    for (int i = 0; i < vector1.length; i++) {
66      vectorSuma[i] = vector1[i] + vector2[i];
67    }
```

```
69    return vectorSuma;
70  }
71
72  /*
73   * Calcula la resta de dos vectores dados
74   */
75  int[] restaVectores(int[] vector1, int[] vector2) {
76    if ((vector1 == null) || (vector2 == null) ||
77        (vector1.length != vector2.length)) {
78      return null;
79    }
80
81    int[] vectorResta = new int[vector1.length];
82    for (int i = 0; i < vector1.length; i++) {
83      vectorResta[i] = vector1[i] - vector2[i];
84    }
85
86    return vectorResta;
87  }
88
89  /*
90   * Calcula la multiplicación de un vector dado por
91   * un escalar dado
92   */
93  int[] multiplicacionPorEscalar(int[] vector1, int escalar) {
94    if (vector1 == null) {
95      return null;
96    }
97
98    int[] nuevoVector = new int[vector1.length];
99    for (int i = 0; i < vector1.length; i++) {
100     nuevoVector[i] = vector1[i] * escalar;
101   }
102
103   return nuevoVector;
104 }
105
106 /*
107  * Calcula el producto escalar de dos vectores
108  */
109 int productoEscalar(int[] vector1, int[] vector2) {
110   int resultado = 0;
111
112   if ((vector1 == null) || (vector2 == null) ||
113       (vector1.length != vector2.length)) {
114     return resultado;
115   }
116
117   int[] nuevoVector = new int[vector1.length];
118   for (int i = 0; i < vector1.length; i++) {
119     resultado += vector1[i] * vector2[i];
120   }
121
122   return resultado;
123 }
```

```
125 void setup(){
126   int longitud = 6;
127   int min = 5;
128   int max = 15;
129   int escalar = 10;

131   int[] miVector1 = crearVectorAleatorio(longitud, min, max);
132   println("Mi primer vector aleatoria de longitud "+longitud+" y
          valores entre "+min+" y "+max+" es:");
133   mostrarVector(miVector1);
134   println("Y su módulo vale: "+moduloVector(miVector1));
135   println();

137   int[] miVector2 = crearVectorAleatorio(longitud, min, max);
138   println("Mi segundo vector aleatoria de longitud "+longitud+" y
          valores entre "+min+" y "+max+" es:");
139   mostrarVector(miVector2);
140   println("Y su módulo vale: "+moduloVector(miVector2));
141   println();

143   int[] vectorSuma = sumaVectores(miVector1, miVector2);
144   println("La suma de ambos vectores es: ");
145   mostrarVector(vectorSuma);
146   println();

148   int[] vectorResta = restaVectores(miVector1, miVector2);
149   println("La resta de ambos vectores es: ");
150   mostrarVector(vectorResta);
151   println();

153   int[] nuevoVector = multiplicacionPorEscalar(miVector1, escalar);
154   println("La multiplicación del primer vector por el escalar "
          +escalar+" es: ");
155   mostrarVector(nuevoVector);
156   println();

158   println("Y el producto escalar de ambos vectores vale: "
          +productoEscalar(miVector1, miVector2));
159 }
```

6.7 Dada una matriz A, implementar las siguientes funciones:

- Calcular la transpuesta A^t
- Indicar si es cuadrada
- Indicar si es simétrica

Haz un programa para mostrar que se cumplen las siguientes propiedades:

- $(A^t)^t = A$ (involutiva)
- $(A+B)^t = A^t + B^t$ (distributiva)
- $(c \cdot A)^t = c \cdot A^t$

```
1 /*
2  * Crea una matriz de un rango dado con valores
3  * aleatorios entre un mínimo y un máximo
4  */
5 int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
      max) {
```

```
6    if ((filas < 1) || (columnas < 1) || (max < min)) {
7      return null;
8    }

10   int[][] matriz = new int[filas][columnas];

12   for (int i = 0; i < filas; i++) {
13     for (int j = 0; j < columnas; j++) {
14       matriz[i][j] = (int)random(min, max+1);
15     }
16   }

18   return matriz;
19 }

21 /*
22  * Suma dos matrices dadas, si es posible.
23  * Si la suma no es posible, devuelve null.
24  */
25 int[][] sumarMatrices(int[][] matriz1, int[][] matriz2) {
26   if ((matriz1 == null) || (matriz2 == null) ||
27       (matriz1.length != matriz2.length) ||
28       (matriz1[0].length != matriz2[0].length)) {
29      return null;
30   }

32   int[][] matrizSuma = new int[matriz1.length][matriz1[0].length];
33   for (int i = 0; i < matriz1.length; i++) {
34     for (int j = 0; j < matriz1[0].length; j++) {
35       matrizSuma[i][j] = matriz1[i][j] + matriz2[i][j];
36     }
37   }

39   return matrizSuma;
40 }

42 /*
43  * Devuelve una matriz multiplicando cada elemento de una
44  * matriz dada por una constante dada.
45  */
46 int[][] productoConstanteMatriz(int[][] matriz, int constante) {
47   if (matriz == null){
48     return null;
49   }

51   int[][] matrizResultado = new int[matriz.length][matriz[0].length];
52   for (int i = 0; i < matriz.length; i++) {
53     for (int j = 0; j < matriz[0].length; j++) {
54       matrizResultado[i][j] = matriz[i][j] * constante;
55     }
56   }
57   return matrizResultado;
58 }

60 /*
61  * Comprueba si dos matrices dadas son iguales
```

```
62   * o no. Si son iguales, devuelve true, y si no
63   * son iguales, devuelve false.
64   */
65  boolean matricesIguales(int[][] matriz1, int[][] matriz2){
66    if ((matriz1 == null) || (matriz2 == null) ||
67        (matriz1.length != matriz2.length) ||
68        (matriz1[0].length != matriz2[0].length)) {
69       return false;
70    }
71
72    for (int i = 0; i < matriz1.length; i++) {
73      for (int j = 0; j < matriz1[0].length; j++) {
74        if (matriz1[i][j] != matriz2[i][j]) {
75          return false;
76        }
77      }
78    }
79
80    return true;
81  }
82
83  /*
84   * Calcula la matriz transpuesta de una matriz dada.
85   */
86  int[][] matrizTranspuesta(int[][] matriz) {
87    if (matriz == null) {
88       return null;
89    }
90
91    int[][] transpuesta = new int[matriz[0].length][matriz.length];
92    for (int i = 0; i < matriz.length; i++) {
93      for (int j = 0; j < matriz[0].length; j++) {
94        transpuesta[j][i] = matriz[i][j];
95      }
96    }
97
98    return transpuesta;
99  }
100
101 /*
102  * Comprueba si una matriz es cuadrada o no.
103  */
104 boolean matrizCuadrada(int[][] matriz){
105   return ((matriz != null) && (matriz.length == matriz[0].length));
106 }
107
108 /*
109  * Comprueba si una matriz es simétrica o no.
110  */
111 boolean matrizSimetrica(int[][] matriz){
112   if ((matriz == null) || !matrizCuadrada(matriz)) {
113     return false;
114   }
115
116   for (int i = 0; i < matriz.length; i++) {
117     for (int j = 0; j < matriz[0].length; j++) {
```

```
118       if (matriz[j][i] != matriz[i][j]) {
119         return false;
120       }
121     }
122   }
123   return true;
124 }


127 /*
128  * Genera y devuelve una representación de tipo
129  * String de una matriz dada.
130  */
131 String matrizAString(int[][] matriz){
132   String cadena = "";

134   if (matriz == null){
135     return cadena;
136   }

138   for (int i = 0; i < matriz.length; i++) {
139     for (int j = 0; j < matriz[0].length; j++) {
140       cadena += matriz[i][j] + "\t";
141     }
142     cadena += "\n";
143   }
144   return cadena;
145 }

147 /*
148  * Muestra por pantalla los valores de una
149  * matriz dada
150  */
151 void mostrarMatriz(int[][] matriz){
152    print(matrizAString(matriz));
153 }

155 void setup(){
156   int filas = 5;
157   int columnas = 5;
158   int min = 5;
159   int max = 15;
160   int constante = 10;

162   int[][] miMatriz1 = crearMatrizAleatoria(filas, columnas, min, max);
163   println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
          columnas, con valores entre "+min+" y "+max+" es:");
164   mostrarMatriz(miMatriz1);
165   println();

167   int[][] miMatriz2 = crearMatrizAleatoria(filas, columnas, min, max);
168   println("Mi matriz aleatoria B con "+filas+" filas y "+columnas+"
          columnas, con valores entre "+min+" y "+max+" es:");
169   mostrarMatriz(miMatriz2);
170   println();
```

```
172    int[][] miMatrizTranspuesta = matrizTranspuesta(miMatriz1);
173    println("La matriz transpuesta de A es:");
174    mostrarMatriz(miMatrizTranspuesta);
175    println();

177    if(matrizCuadrada(miMatriz1)) {
178       println("La matriz A sí es cuadrada\n");
179    } else {
180       println("La matriz A no es cuadrada\n");
181    }

183    if(matrizSimetrica(miMatriz1)) {
184       println("La matriz A sí es simétrica\n");
185    } else {
186       println("La matriz A no es simétrica\n");
187    }

189    if(matricesIguales(miMatriz1,
           matrizTranspuesta(miMatrizTranspuesta))) {
190       println("La matriz A sí cumple la propiedad involutiva\n");
191    } else {
192       println("La matriz A no cumple la propiedad involutiva\n");
193    }

195    if(matricesIguales(
196        matrizTranspuesta(sumarMatrices(miMatriz1, miMatriz2)),
197        sumarMatrices(matrizTranspuesta(miMatriz1),
               matrizTranspuesta(miMatriz2)))) {
198       println("Las matrices A y B sí cumplen la propiedad
              distributiva\n");
199    } else {
200       println("Las matrices A y B sí cumplen la propiedad
              distributiva\n");
201    }

203    if(matricesIguales(
204        matrizTranspuesta(productoConstanteMatriz(miMatriz1,
               constante)),
205        productoConstanteMatriz(matrizTranspuesta(miMatriz1),
               constante))) {
206       println("La transpuesta de multiplicar "+constante+" por A sí es
              igual a la multiplicación de "+constante+" por la transpuesta
              de A\n");
207    } else {
208       println("La transpuesta de multiplicar "+constante+" por A no es
              igual a la multiplicación de "+constante+" por la transpuesta
              de A\n");
209    }
210 }
```

6.8 Dada una matriz A, implementar las siguientes funciones:

- Dado un índice i de una fila, eliminar la fila correspondiente de la matriz A.
- Dado un índice j de una columna, eliminar la columna correspondiente de la matriz A.
- Dado un índice i de una fila y una nueva fila, añadir esa nueva fila en el índice dado de la matriz A.

- Dado un índice j de una columna y una nueva columna, añadir esa nueva columna en el índice dado de la matriz A.
- Desplazar las filas de la matriz A hacia abajo, de forma circular, un número dado de posiciones.
- Desplazar las columnas de la matriz A hacia la derecha, de forma circular, un número dado de posiciones.

Nota: Todas las operaciones deberán realizarse sin modificar la matriz original A.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }
9
10   int[] lista = new int[longitud];
11
12   for (int i = 0; i < longitud; i++) {
13      lista[i] = (int)random(min, max+1);
14   }
15
16   return lista;
17 }
18
19 /*
20  * Crea una matriz de un rango dado con valores
21  * aleatorios entre un mínimo y un máximo
22  */
23 int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
       max) {
24   if ((filas < 1) || (columnas < 1) || (max < min)) {
25     return null;
26   }
27
28   int[][] matriz = new int[filas][columnas];
29
30   for (int i = 0; i < filas; i++) {
31     for (int j = 0; j < columnas; j++) {
32       matriz[i][j] = (int)random(min, max+1);
33     }
34   }
35
36   return matriz;
37 }
38
39 /*
40  * Elimina la fila del índice dado en la
41  * matriz dada
42  */
43 int[][] eliminarFila(int[][] array, int indiceFila) {
44   int filas = array.length;
45   int columnas = array[0].length;
46   if (indiceFila < 0 || indiceFila >= filas) {
```

```
47      return null;
48    }
49    int[][] nuevoArray = new int[filas - 1][columnas];
50    for (int i = 0, j = 0; i < filas; i++) {
51      if (i != indiceFila) {
52        nuevoArray[j] = array[i];
53        j++;
54      }
55    }
56    return nuevoArray;
57  }

59  /*
60   * Elimina la columna del índice dado en la
61   * matriz dada
62   */
63  int[][] eliminarColumna(int[][] matriz, int indiceColumna) {
64    int[][] resultado = new int[matriz.length][matriz[0].length - 1];
65    for (int i = 0; i < matriz.length; i++) {
66      int contador = 0;
67      for (int j = 0; j < matriz[i].length; j++) {
68        if (j != indiceColumna) {
69          resultado[i][contador] = matriz[i][j];
70          contador++;
71        }
72      }
73    }
74    return resultado;
75  }

77  /*
78   * Añade la fila dada en el índice dado de la
79   * matriz dada
80   */
81  int[][] anadirFila(int[][] matriz, int[] fila, int indiceFila) {
82    int[][] resultado = new int[matriz.length + 1][matriz[0].length];
83    int contador = 0;
84    for (int i = 0; i < resultado.length; i++) {
85      if (i == indiceFila) {
86        resultado[i] = fila;
87      } else {
88        resultado[i] = matriz[contador];
89        contador++;
90      }
91    }
92    return resultado;
93  }

95  /*
96   * Añade la columna dada en el índice dado de la
97   * matriz dada
98   */
99  int[][] anadirColumna(int[][] matriz, int[] columna, int
        indiceColumnaAAgregar) {
100   int[][] resultado = new int[matriz.length][matriz[0].length + 1];
101   for (int i = 0; i < resultado.length; i++) {
```

```
102       int contador = 0;
103       for (int j = 0; j < resultado[0].length; j++) {
104         if (j == indiceColumnaAAgregar) {
105           resultado[i][j] = columna[i];
106         } else {
107           resultado[i][j] = matriz[i][contador];
108           contador++;
109         }
110       }
111     }
112     return resultado;
113   }
114
115   /*
116    * Desplaza las filas de la matriz dada hacia
117    * abajo circularmente, un número dado de posiciones
118    */
119
120   int[][] desplazarFilasCircular(int[][] matriz, int desplazamiento) {
121     int[][] resultado = new int[matriz.length][matriz[0].length];
122     for (int i = 0; i < matriz.length; i++) {
123       for (int j = 0; j < matriz[0].length; j++) {
124         resultado[(i + desplazamiento) % matriz.length][j] =
              matriz[i][j];
125       }
126     }
127     return resultado;
128   }
129
130   /*
131    * Desplaza las columnas de la matriz dada hacia
132    * la derecha circularmente, un número dado de posiciones
133    */
134   int[][] desplazarColumnasCircular(int[][] matriz, int desplazamiento)
        {
135     int[][] resultado = new int[matriz.length][matriz[0].length];
136     for (int i = 0; i < matriz.length; i++) {
137       for (int j = 0; j < matriz[0].length; j++) {
138         resultado[i][(j + desplazamiento) % matriz[0].length] =
              matriz[i][j];
139       }
140     }
141     return resultado;
142   }
143
144
145   /*
146    * Genera y devuelve una representación de tipo
147    * String de una matriz dada.
148    */
149   String matrizAString(int[][] matriz){
150     String cadena = "";
151
152     if (matriz == null){
153       return cadena;
154     }
```

```
156   for (int i = 0; i < matriz.length; i++) {
157     for (int j = 0; j < matriz[0].length; j++) {
158       cadena += matriz[i][j] + "\t";
159     }
160     cadena += "\n";
161   }
162   return cadena;
163 }

165 /*
166  * Muestra por pantalla los valores de una
167  * matriz dada
168  */
169 void mostrarMatriz(int[][] matriz){
170    print(matrizAString(matriz));
171 }

173 /*
174  * Muestra por pantalla los valores de una
175  * lista dada
176  */
177 void mostrarLista(int[] lista){
178    if (lista != null) {
179      for (int i = 0; i < lista.length; i++){
180        print(lista[i]+"\t");
181      }
182    }
183    println();
184 }

186 void setup(){
187   int filas = 4;
188   int columnas = 5;
189   int min = 5;
190   int max = 15;
191   int N = 5;

193   for (int i = 0; i < N; i++) {
194     int[][] miMatriz1 = crearMatrizAleatoria(filas, columnas, min,
          max);
195     println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
          columnas, con valores entre "+min+" y "+max+" es:");
196     mostrarMatriz(miMatriz1);
197     println();

199     int filaAEliminar = (int) random(filas);
200     miMatriz1 = eliminarFila(miMatriz1, filaAEliminar);
201     println("Eliminamos la fila "+filaAEliminar);
202     mostrarMatriz(miMatriz1);
203     println();

205     int columnaAEliminar = (int) random(columnas);
206     miMatriz1 = eliminarColumna(miMatriz1, columnaAEliminar);
207     println("Eliminamos la columna "+columnaAEliminar);
208     mostrarMatriz(miMatriz1);
```

```
209     println();

211     int[] fila = crearListaAleatoria(miMatriz1[0].length, min, max);
212     int indiceFila = (int)random(miMatriz1[0].length);
213     println("Insertamos la siguiente fila en el índice "+indiceFila);
214     mostrarLista(fila);
215     println();

217     miMatriz1 = anadirFila(miMatriz1, fila, indiceFila);
218     mostrarMatriz(miMatriz1);
219     println();

221     int[] columna = crearListaAleatoria(miMatriz1.length, min, max);
222     int indiceColumna = (int)random(miMatriz1.length);
223     println("Insertamos la siguiente columna en el índice "
            +indiceColumna);
224     mostrarLista(columna);
225     println();

227     miMatriz1 = anadirColumna(miMatriz1, columna, indiceColumna);
228     mostrarMatriz(miMatriz1);
229     println();

231     int desplazamiento = (int)random(miMatriz1[0].length);
232     println("Desplazamos las filas de la matriz en "+desplazamiento+"
            posiciones");
233     miMatriz1 = desplazarFilasCircular(miMatriz1, desplazamiento);
234     mostrarMatriz(miMatriz1);
235     println();

237     desplazamiento = (int)random(miMatriz1.length);
238     println("Desplazamos las columnas de la matriz en "
            +desplazamiento+" posiciones");
239     miMatriz1 = desplazarColumnasCircular(miMatriz1, desplazamiento);
240     mostrarMatriz(miMatriz1);
241     println();
242   }
243 }
```

6.9 A partir del ejercicio **6.4**, crea un registro para representar una Fecha y utilízalo dentro del registro de Alumno. Crea las siguientes funciones:

- Dadas dos fechas, calcula la diferencia entre ambas expresada en días. Asume meses de 30 días y sin años bisiestos.
- Dado un número de días, calcular su equivalente en número de años, meses y días.
- Dado un alumno, calcular su edad. Añade este campo al Alumno.
- Generaliza para que un Alumno pueda tener de 1 a n Asignaturas.
- Añade un campo con la nota media de todas las asignaturas de un alumno.

6.10 Crea un registro para representar un punto $p = (x, y)$ y otro registro para representar un círculo c, el cual viene determinado por su centro $p = (p_1, p_2)$ y un radio $r \in \mathbb{R}$. A partir de dichos registros, crea las siguientes funciones:

- Obtener un punto aleatorio cuyas coordenadas toman valores dentro de un rango $[min, max]$.
- Dados dos puntos p_1 y p_2, calcular la distancia euclídea entre ambos.

- Dados dos puntos p_1 y p_2, calcular el punto medio p_m.
- Obtener un círculo aleatorio donde las coordenadas de su centro toman valores dentro de un rango $[min_c, max_c]$ y cuyo radio toma valores dentro de un rango $[min_r, max_r]$
- Dados dos círculos c_1 y c_2, indicar si se solapan o no.

```
1  class Punto {
2    float coordX;
3    float coordY;
4  }

6  class Circulo {
7    Punto centro;
8    float radio;
9  }

11 /*
12  * Genera un punto aleatorio con coordenadas
13  * entre un mínimo y un máximo dados
14  */
15 Punto puntoAleatorio(float min, float max){
16   Punto punto = new Punto();
17   punto.coordX = random(min, max);
18   punto.coordY = random(min, max);
19   return punto;
20 }

22 /*
23  * Calcula la distancia euclídea entr dos
24  * puntos dados
25  */
26 float distanciaEuclidea(Punto punto1, Punto punto2) {
27   return sqrt(pow(punto1.coordX-punto2.coordX, 2) +
         pow(punto1.coordY-punto2.coordY, 2));
28 }

30 /*
31  * Calcula el punto medio entre dos
32  * puntos dados
33  */
34 Punto puntoMedio(Punto punto1, Punto punto2) {
35   Punto puntoMedio = new Punto();

37   puntoMedio.coordX = (punto1.coordX+punto2.coordX)/2.0;
38   puntoMedio.coordY = (punto1.coordY+punto2.coordY)/2.0;

40   return puntoMedio;
41 }

43 /*
44  * Genera un círculo aleatorio cuyo centro tiene coordenadas
45  * entre un mínimo y un máximo dados y cuyo radio toma un
46  * valor aleatorio entre un mínimo y un máximo
47  */
48 Circulo circuloAleatorio(float minCentro, float maxCentro, float
      minRadio, float maxRadio) {
49   Circulo circulo = new Circulo();
```

```
50    circulo.centro = puntoAleatorio(minCentro, maxCentro);
51    circulo.radio = random(minRadio, maxRadio);
52    return circulo;
53  }

55  /*
56   * Indica si dos círculos dados se solapan o no
57   */
58  boolean circulosSolapan(Circulo circulo1, Circulo circulo2) {
59    float distanciaCentros = distanciaEuclidea(circulo1.centro,
          circulo2.centro);
60    return (distanciaCentros <= circulo1.radio+circulo2.radio) &&
61           (distanciaCentros >= abs(circulo1.radio-circulo2.radio));
62  }

64  /*
65   * Genera la representación en String de un círculo dado
66   */
67  String puntoAString(Punto punto) {
68    return "("+punto.coordX+", "+punto.coordY+")";
69  }

71  void setup(){
72    float min = 0;
73    float max = 50;
74    int N = 20;

76    for (int i = 0; i < N; i++) {
77      Punto punto1 = puntoAleatorio(min, max);
78      Punto punto2 = puntoAleatorio(min, max);
79      float distancia = distanciaEuclidea(punto1, punto2);
80      println("La distancia entre "+puntoAString(punto1)+" y "
            +puntoAString(punto2)+" es "+distancia);

82      Circulo circulo1 = circuloAleatorio(min, max, min, max);
83      Circulo circulo2 = circuloAleatorio(min, max, min, max);
84      print("El círculo con centro "+puntoAString(circulo1.centro)+" y
            radio "+circulo1.radio+
85          " y el círculo con centro "+puntoAString(circulo2.centro)+" y
              radio "+circulo2.radio);
86      if (circulosSolapan(circulo1, circulo2)) {
87        println(" sí se solapan");
88      } else {
89        println(" no se solapan");
90      }
91      println();
92    }
93  }
```

6.3 Tema 6: Ejercicios para casa

6.11 Dado un array A de enteros, implementar las siguientes funciones:

- Devolver otro array B en orden inverso.
- Dado otro array B, devolver la unión de ambos $C = A \cup B$
- Dado otro array B, devolver la intersección de ambos $C = A \cap B$

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }
9
10   int[] lista = new int[longitud];
11
12   for (int i = 0; i < longitud; i++) {
13      lista[i] = (int)random(min, max+1);
14   }
15
16   return lista;
17 }
18
19 /*
20  * Dada una lista crea una lista inversa.
21  */
22 int[] listaInversa(int[] lista) {
23   if (lista == null) {
24      return null;
25   }
26
27   int[] inversa = new int[lista.length];
28   for (int i = 0; i < lista.length; i++) {
29      inversa[lista.length-1-i] = lista[i];
30   }
31
32   return inversa;
33 }
34
35 /*
36  * Dadas dos listas, calcula la unión de ambas.
37  */
38 int[] unionListas(int[] lista1, int[] lista2){
39    if ((lista1 == null) || (lista2 == null)){
40      return null;
41    }
42
43    int[] union = new int[lista1.length + lista2.length];
44    int cursor = 0;
45    for (int i = 0; i < lista1.length; i++){
46       if (!buscarValor(union, lista1[i])) {
47         union[cursor] = lista1[i];
48         cursor++;
49       }
```

```
50      }
51
52      for (int i = 0; i < lista2.length; i++){
53         if (!buscarValor(union, lista2[i])) {
54           union[cursor] = lista2[i];
55           cursor++;
56         }
57      }
58
59      union = new int[cursor];
60
61      cursor = 0;
62      for (int i = 0; i < lista1.length; i++){
63         if (!buscarValor(union, lista1[i])) {
64           union[cursor] = lista1[i];
65           cursor++;
66         }
67      }
68
69      for (int i = 0; i < lista2.length; i++){
70         if (!buscarValor(union, lista2[i])) {
71           union[cursor] = lista2[i];
72           cursor++;
73         }
74      }
75
76      return union;
77  }
78
79  /*
80   * Dadas dos listas, calcula la intersección de ambas.
81   */
82  int[] interseccionListas(int[] lista1, int[] lista2){
83      if ((lista1 == null) || (lista2 == null)){
84        return null;
85      }
86
87      int contador = 0;
88      int[] interseccion = new int[lista1.length];
89      for (int i = 0; i < lista1.length; i++){
90         if (buscarValor(lista2, lista1[i])
91         && !buscarValor(interseccion, lista1[i])) {
92           interseccion[contador] = lista1[i];
93           contador++;
94         }
95      }
96
97      interseccion = new int[contador];
98      contador = 0;
99      for (int i = 0; i < lista1.length; i++){
100        if (buscarValor(lista2, lista1[i])
101        && !buscarValor(interseccion, lista1[i])) {
102          interseccion[contador] = lista1[i];
103          contador++;
104        }
105     }
```

```
107     return interseccion;
108 }



112 /*
113  * Busca un valor dado en una lista dada.
114  * Devuelve true si encuentra el valor y
115  * false si no lo encuentra
116  */
117 boolean buscarValor(int[] lista, int valor){
118     if (lista != null) {
119       for (int i = 0; i < lista.length; i++){
120          if (lista[i] == valor) {
121             return true;
122          }
123       }
124     }
125     return false;
126 }

128 /*
129  * Muestra por pantalla los valores de una
130  * lista dada
131  */
132 void mostrarLista(int[] lista){
133     if (lista != null) {
134       for (int i = 0; i < lista.length; i++){
135         print(lista[i]+" ");
136       }
137     }
138     println();
139 }

141 void setup(){
142   int longitud = 10;
143   int min = 5;
144   int max = 15;
145   int nuevoValor = 10;

147   int[] miLista1 = crearListaAleatoria(longitud, min, max);
148   println("Mi lista A aleatoria de longitud "+longitud+" y valores
          entre "+min+" y "+max+" es:");
149   mostrarLista(miLista1);
150   println();

152   int[] miLista2 = crearListaAleatoria(longitud, min, max);
153   println("Mi lista B aleatoria de longitud "+longitud+" y valores
          entre "+min+" y "+max+" es:");
154   mostrarLista(miLista2);
155   println();

157   println("La lista inversa de A es:");
158   int[] miListaInversa1 = listaInversa(miLista1);
159   mostrarLista(miListaInversa1);
```

```
160   println();

162   println("La lista inversa de B es:");
163   int[] miListaInversa2 = listaInversa(miLista2);
164   mostrarLista(miListaInversa2);
165   println();

167   int[] union = unionListas(miLista1, miLista2);
168   println("La unión de las listas A y B es:");
169   mostrarLista(union);
170   println();

172   int[] interseccion = interseccionListas(miLista1, miLista2);
173   println("La intersección de las listas A y B es:");
174   mostrarLista(interseccion);
175   println();

177 }
```

6.12 Dado un número natural obtén:

- La lista de divisores propios de dicho número.
- La lista de factores primos distintos de dicho número.

```
1  /*
2   * Calcula los divisores propios de un
3   * número dado.
4   */
5  int[] divisoresPropios(int numero) {
6     int numDivisores = 0;

8     for (int i = numero; i > 1; i--) {
9       if (numero%i == 0) {
10        numDivisores++;
11      }
12    }

14    int[] divisores = new int[numDivisores];
15    numDivisores = 0;
16    for (int i = numero; i > 1; i--) {
17      if (numero%i == 0) {
18        divisores[numDivisores] = i;
19        numDivisores++;
20      }
21    }

23    return divisores;
24 }

26 /*
27  * Indica si un número dado es primo o no.
28  */
29 boolean esPrimo(int numero) {
30    if (numero < 1) {
31      return false;
32    }
```

```
34    for (int i = numero-1; i > 1; i--) {
35      if (numero%i == 0) {
36        return false;
37      }
38    }
39    return true;
40  }

42  /*
43   * Calcula los factores primos de un
44   * número dado.
45   */
46  int[] factoresPrimos(int numero) {
47    int numFactoresPrimos = 0;

49    for (int i = numero; i > 1; i--) {
50      if ((numero%i == 0) && esPrimo(i)) {
51        numFactoresPrimos++;
52      }
53    }

55    int[] factoresPrimos = new int[numFactoresPrimos];
56    numFactoresPrimos = 0;
57    for (int i = numero; i > 1; i--) {
58      if ((numero%i == 0) && esPrimo(i)) {
59        factoresPrimos[numFactoresPrimos] = i;
60        numFactoresPrimos++;
61      }
62    }

64    return factoresPrimos;
65  }


68  /*
69   * Muestra por pantalla los valores de una
70   * lista dada
71   */
72  void mostrarLista(int[] lista){
73    if (lista != null) {
74      for (int i = 0; i < lista.length; i++){
75        print(lista[i]+" ");
76      }
77    }
78    println();
79  }

81  void setup(){
82    int min = 50;
83    int max = 200;
84    int N = 10;

86    for (int i = 0; i < N; i++) {
87      int numero = (int)random(min, max+1);
88      int[] divisores = divisoresPropios(numero);
89      int[] primos = factoresPrimos(numero);
```

```
91      print("Los divisores propios del "+numero+" son: ");
92      mostrarLista(divisores);

94      print("Los factores primos del "+numero+" son: ");
95      mostrarLista(primos);
96      println();
97    }
98  }
```

6.13 Obtener la lista de números primos que están dentro de un rango de números naturales.

```
1  /*
2   * Indica si un número dado es primo o no.
3   */
4  boolean esPrimo(int numero) {
5     if (numero < 1) {
6       return false;
7     }

9     for (int i = numero-1; i > 1; i--) {
10      if (numero%i == 0) {
11        return false;
12      }
13    }
14    return true;
15 }

17 /*
18  * Calcula los números primos dentro de un
19  * rango dado.
20  */
21 int[] primosEnRango(int min, int max) {
22    int numPrimos = 0;

24    for (int i = min; i <= max; i++) {
25      if (esPrimo(i)) {
26        numPrimos++;
27      }
28    }

30    int[] primos = new int[numPrimos];
31    numPrimos = 0;
32    for (int i = min; i <= max; i++) {
33      if (esPrimo(i)) {
34        primos[numPrimos] = i;
35        numPrimos++;
36      }
37    }

39    return primos;
40 }


43 /*
44  * Muestra por pantalla los valores de una
45  * lista dada
```

```
 */
void mostrarLista(int[] lista){
   if (lista != null) {
     for (int i = 0; i < lista.length; i++){
       print(lista[i]+" ");
     }
   }
   println();
}

void setup(){
  int min = 1;
  int max = 500;
  int N = 4;

  for (int i = 0; i < N; i++) {
    int _min = (int) random(min, max+1);
    int _max = _min + (int) random(max-_min, max+1);
    int[] primos = primosEnRango(_min, _max);

    print("Los números primos dentro del rango ["+_min+", "+_max+"]
        son: ");
    mostrarLista(primos);
    println();
  }
}
```

6.14 Obtener la lista de k números primos que son inferiores a un número dado. Justifica qué hacer cuando no se alcancen los k.

```
/*
 * Indica si un número dado es primo o no.
 */
boolean esPrimo(int numero) {
   if (numero < 1) {
     return false;
   }

   for (int i = numero-1; i > 1; i--) {
     if (numero%i == 0) {
       return false;
     }
   }
   return true;
}

/*
 * Retorna los k números primos inferiores a
 * un número dado. Si el número de primos
 * inferiores a dicho número es menor que k,
 * entonces se devuelven esos primos.
 */
int[] kPrimosInferiores(int numero, int k) {
   if ((numero < 1) || (k < 1)){
     return null;
   }
   int numPrimos = 0;
```

```
    int[] primos = new int[k];
    for (int i = numero-1; (i > 1) && (numPrimos < k); i--) {
      if (esPrimo(i)) {
        primos[numPrimos] = i;
        numPrimos++;
      }
    }

    if (numPrimos < k) {
       int[] primosAux = new int[numPrimos];
       for (int i = 0; i < numPrimos; i++) {
         primosAux[i] = primos[i];
       }
       return primosAux;
    }

    return primos;
}


/*
 * Muestra por pantalla los valores de una
 * lista dada
 */
void mostrarLista(int[] lista){
    if (lista != null) {
      for (int i = 0; i < lista.length; i++){
        print(lista[i]+" ");
      }
    }
    println();
}

void setup(){
  int min = 1;
  int max = 500;
  int N = 10;

  for (int i = 0; i < N; i++) {
    int numero = (int) random(min, max+1);
    int k = (int) random(min/10, (max+1)/10);
    int[] primos = kPrimosInferiores(numero, k);

    print("Los "+k+" números primos inferiores a "+numero+" son: ");
    mostrarLista(primos);
    println();
  }
}
```

6.15 Dada una matriz de $n \times m$ valores reales, construye un función que los retorna en un array1D con todos los valores de la matriz por filas.

Por ejemplo, la transformación de $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3}$ es $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}_{1\times6}$

```
/*
```

```
2   * Crea una matriz de un rango dado con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
       max) {
6    if ((filas < 1) || (columnas < 1) || (max < min)) {
7      return null;
8    }

10   int[][] matriz = new int[filas][columnas];

12   for (int i = 0; i < filas; i++) {
13     for (int j = 0; j < columnas; j++) {
14       matriz[i][j] = (int)random(min, max+1);
15     }
16   }

18   return matriz;
19 }

21 /*
22  * Transforma una matriz dada en un array1D.
23  */
24 int[] matrizAArray1D(int[][] matriz) {
25   if (matriz == null) {
26      return null;
27   }

29   int[] array1D = new int[matriz.length*matriz[0].length];
30   for (int i = 0; i < matriz.length; i++) {
31     for (int j = 0; j < matriz[0].length; j++) {
32       array1D[(i*matriz.length)+j] = matriz[i][j];
33     }
34   }

36   return array1D;
37 }

39 /*
40  * Genera y devuelve una representación de tipo
41  * String de una matriz dada.
42  */
43 String matrizAString(int[][] matriz){
44   String cadena = "";

46   if (matriz == null){
47     return cadena;
48   }

50   for (int i = 0; i < matriz.length; i++) {
51     for (int j = 0; j < matriz[0].length; j++) {
52       cadena += matriz[i][j] + "\t";
53     }
54     cadena += "\n";
55   }
56   return cadena;
```

```
57 }

59 /*
60  * Muestra por pantalla los valores de una
61  * matriz dada
62  */
63 void mostrarMatriz(int[][] matriz){
64    print(matrizAString(matriz));
65 }

67 /*
68  * Muestra por pantalla los valores de una
69  * lista dada
70  */
71 void mostrarLista(int[] lista){
72    if (lista != null) {
73      for (int i = 0; i < lista.length; i++){
74        print(lista[i]+" ");
75      }
76    }
77    println();
78 }

80 void setup(){
81   int filas = 5;
82   int columnas = 5;
83   int min = 5;
84   int max = 15;
85   int N = 4;

87   for (int i = 0; i < N; i++) {
88     int[][] miMatriz1 = crearMatrizAleatoria(filas, columnas, min,
         max);
89     println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
         columnas, con valores entre "+min+" y "+max+" es:");
90     mostrarMatriz(miMatriz1);
91     println();

93     int[] array1D = matrizAArray1D(miMatriz1);
94     println("El array1D de la matriz A es:");
95     mostrarLista(array1D);
96     println();
97   }
98 }
```

6.16 A partir del ejercicio **6.9**, crea el registro Clase con al menos los siguientes campos: curso, conjunto de alumnos.

Dada una Clase, Haz las siguientes funciones:

- Devolver el alumno con la máxima nota media de todas sus asignaturas.
- Dada una clase y el nombre de una asignatura, devolver el alumno con la máxima nota para dicha asignatura.
- Dada una clase y el nombre de una asignatura, devolver las notas de todos los alumnos para esa asignatura.
- Devolver la media de edad de la clase.

Crea una Clase con 5 Alumnos, cada uno de ellos con 3 Asignaturas y prueba las funciones anteriores.

```
class Clase {
  int curso;
  Alumno[] alumnos;
}

class Fecha {
  int dia;
  int mes;
  int anyo;
}

class Edad {
  int anyos;
  int meses;
  int dias;
}

class Asignatura {
  String nombre;
  int creditos;
  int cuatrimestre;
  float calificacion;
}

class Alumno {
  String nombre;
  String apellidos;
  Fecha fechaNacimiento;
  Edad edad;
  boolean sexo; //True para mujer y false para hombre
  String email;

  Asignatura[] asignaturas;
  float notaMedia;
}

/*
 * Calcula la diferencia entre dos fechas dadas
 * fecha1 y fecha2, expresada en días.
 * Si fecha2 es posterior a fecha1, entonces
 * devuelve -1 como código de error.
 */
int diferenciaFechas(Fecha fecha1, Fecha fecha2) {
  int numDiasFecha1 = fecha1.anyo*365 + (fecha1.mes-1)*30 +
      fecha1.dia;
  int numDiasFecha2 = fecha2.anyo*365 + (fecha2.mes-1)*30 +
      fecha2.dia;

  if (numDiasFecha2 > numDiasFecha1) {
    return -1;
  }

  return numDiasFecha1 - numDiasFecha2;
}
```

```
54 /*
55  * Calcula una edad (años, meses y días)
56  * a partir de un número total de días
57  */
58 Edad calcularEdad(int numeroDias) {
59   Edad edad = new Edad();

61   edad.anyos = numeroDias / 365;
62   edad.meses = (numeroDias % 365) / 30;
63   edad.dias = (numeroDias % 365) % 30;

65   return edad;
66 }

68 /*
69  * Convierte una edad en su equivalente
70  * en días.
71  */
72 int edadADias(Edad edad) {
73    return edad.anyos*365 + (edad.meses-1)*30 + edad.dias;
74 }

76 /*
77  * Devuelve el alumno con la máxima nota
78  * media de una clase dada.
79  */
80 Alumno mejorAlumnoClase(Clase clase) {
81   float mejorNota = -1;
82   Alumno mejorAlumno = null;

84   for (int i = 0; i < clase.alumnos.length; i++) {
85     Alumno alumno = clase.alumnos[i];
86     if (alumno.notaMedia > mejorNota) {
87       mejorNota = alumno.notaMedia;
88       mejorAlumno = alumno;
89     }
90   }
91   return mejorAlumno;
92 }

94 /*
95  * Devuelve el alumno con la máxima nota
96  * media de una clase dada, para una
97  * asignatura dada.
98  */
99 Alumno mejorAlumnoAsignatura(Clase clase, String nombreAsignatura) {
100   float mejorNota = -1;
101   Alumno mejorAlumno = null;

103   for (int i = 0; i < clase.alumnos.length; i++) {
104     Alumno alumno = clase.alumnos[i];
105     for (int j = 0; j < alumno.asignaturas.length; j++) {
106       Asignatura asignatura = alumno.asignaturas[j];
107       if ((asignatura.nombre.equals(nombreAsignatura))
108        &&( asignatura.calificacion > mejorNota)) {
```

```
109         mejorNota = asignatura.calificacion;
110         mejorAlumno = alumno;
111       }
112     }
113   }
114   return mejorAlumno;
115 }

117 /*
118  * Devuelve un array con las notas de los
119  * alumnos de una clase dada, para una
120  * asignatura dada.
121  */
122 float[] notasAsignatura(Clase clase, String nombreAsignatura) {
123   int numNotas = 0;

125   for (int i = 0; i < clase.alumnos.length; i++) {
126     Alumno alumno = clase.alumnos[i];
127     for (int j = 0; j < alumno.asignaturas.length; j++) {
128       Asignatura asignatura = alumno.asignaturas[j];
129       if (asignatura.nombre.equals(nombreAsignatura)) {
130         numNotas++;
131       }
132     }
133   }

135   float[] notas = new float[numNotas];
136   numNotas = 0;

138   for (int i = 0; i < clase.alumnos.length; i++) {
139     Alumno alumno = clase.alumnos[i];
140     for (int j = 0; j < alumno.asignaturas.length; j++) {
141       Asignatura asignatura = alumno.asignaturas[j];
142       if (asignatura.nombre.equals(nombreAsignatura)) {
143         notas[numNotas] = asignatura.calificacion;
144         numNotas++;
145       }
146     }
147   }

149   return notas;
150 }

152 /*
153  * Calcula la edad media de los
154  * alumnos de una clase dada.
155  */
156 Edad mediaEdadClase(Clase clase) {
157   int mediaEdadEnDias = 0;
158   for (int i = 0; i < clase.alumnos.length; i++) {
159     Alumno alumno = clase.alumnos[i];
160     mediaEdadEnDias += edadADias(alumno.edad);
161   }
162   mediaEdadEnDias = mediaEdadEnDias / clase.alumnos.length;

164   return calcularEdad(mediaEdadEnDias);
```

```
165 }

167 void setup() {
168   Clase clase = new Clase();
169   clase.curso = 1;
170   clase.alumnos = new Alumno[5];

172   Fecha hoy = new Fecha();
173   hoy.anyo = 2022;
174   hoy.mes = 3;
175   hoy.dia = 23;

177   Alumno alberto = new Alumno();
178   Alumno manuel = new Alumno();
179   Alumno sofia = new Alumno();
180   Alumno pedro = new Alumno();
181   Alumno marina = new Alumno();

183   clase.alumnos[0] = alberto;
184   clase.alumnos[1] = manuel;
185   clase.alumnos[2] = sofia;
186   clase.alumnos[3] = pedro;
187   clase.alumnos[4] = marina;

189   alberto.nombre = "Alberto";
190   alberto.apellidos = "García Martínez";
191   alberto.fechaNacimiento = new Fecha();
192   alberto.fechaNacimiento.anyo = 2001;
193   alberto.fechaNacimiento.mes = 7;
194   alberto.fechaNacimiento.dia = 20;
195   alberto.edad = calcularEdad(diferenciaFechas(hoy,
          alberto.fechaNacimiento));
196   alberto.sexo = false;
197   alberto.email = "alberto.garcia-martinez@um.es";
198   alberto.notaMedia = 0.0;

200   alberto.asignaturas = new Asignatura[3];
201   alberto.asignaturas[0] = new Asignatura();
202   alberto.asignaturas[0].nombre = "Programación";
203   alberto.asignaturas[0].creditos = 6;
204   alberto.asignaturas[0].cuatrimestre = 2;
205   alberto.asignaturas[0].calificacion = random(0, 10);

207   alberto.asignaturas[1] = new Asignatura();
208   alberto.asignaturas[1].nombre = "Álgebra";
209   alberto.asignaturas[1].creditos = 6;
210   alberto.asignaturas[1].cuatrimestre = 1;
211   alberto.asignaturas[1].calificacion = random(0, 10);

213   alberto.asignaturas[2] = new Asignatura();
214   alberto.asignaturas[2].nombre = "Matemática discreta";
215   alberto.asignaturas[2].creditos = 6;
216   alberto.asignaturas[2].cuatrimestre = 2;
217   alberto.asignaturas[2].calificacion = random(0, 10);

219   manuel.nombre = "Manuel";
```

```
220   manuel.apellidos = "Buitrago Gil";
221   manuel.fechaNacimiento = new Fecha();
222   manuel.fechaNacimiento.anyo = 2001;
223   manuel.fechaNacimiento.mes = 12;
224   manuel.fechaNacimiento.dia = 2;
225   manuel.edad = calcularEdad(diferenciaFechas(hoy,
          manuel.fechaNacimiento));
226   manuel.sexo = false;
227   manuel.email = "manuel.bg@um.es";
228   manuel.notaMedia = 0.0;

230   manuel.asignaturas = new Asignatura[3];
231   manuel.asignaturas[0] = new Asignatura();
232   manuel.asignaturas[0].nombre = "Programación";
233   manuel.asignaturas[0].creditos = 6;
234   manuel.asignaturas[0].cuatrimestre = 2;
235   manuel.asignaturas[0].calificacion = random(0, 10);

237   manuel.asignaturas[1] = new Asignatura();
238   manuel.asignaturas[1].nombre = "Álgebra";
239   manuel.asignaturas[1].creditos = 6;
240   manuel.asignaturas[1].cuatrimestre = 1;
241   manuel.asignaturas[1].calificacion = random(0, 10);

243   manuel.asignaturas[2] = new Asignatura();
244   manuel.asignaturas[2].nombre = "Matemática discreta";
245   manuel.asignaturas[2].creditos = 6;
246   manuel.asignaturas[2].cuatrimestre = 2;
247   manuel.asignaturas[2].calificacion = random(0, 10);

249   sofia.nombre = "Sofía";
250   sofia.apellidos = "López Pérez";
251   sofia.fechaNacimiento = new Fecha();
252   sofia.fechaNacimiento.anyo = 2002;
253   sofia.fechaNacimiento.mes = 5;
254   sofia.fechaNacimiento.dia = 18;
255   sofia.edad = calcularEdad(diferenciaFechas(hoy,
          sofia.fechaNacimiento));
256   sofia.sexo = true;
257   sofia.email = "sofialp@um.es";
258   sofia.notaMedia = 0.0;

260   sofia.asignaturas = new Asignatura[3];
261   sofia.asignaturas[0] = new Asignatura();
262   sofia.asignaturas[0].nombre = "Programación";
263   sofia.asignaturas[0].creditos = 6;
264   sofia.asignaturas[0].cuatrimestre = 2;
265   sofia.asignaturas[0].calificacion = random(0, 10);

267   sofia.asignaturas[1] = new Asignatura();
268   sofia.asignaturas[1].nombre = "Álgebra";
269   sofia.asignaturas[1].creditos = 6;
270   sofia.asignaturas[1].cuatrimestre = 1;
271   sofia.asignaturas[1].calificacion = random(0, 10);

273   sofia.asignaturas[2] = new Asignatura();
```

```
274   sofia.asignaturas[2].nombre = "Matemática discreta";
275   sofia.asignaturas[2].creditos = 6;
276   sofia.asignaturas[2].cuatrimestre = 2;
277   sofia.asignaturas[2].calificacion = random(0, 10);

279   pedro.nombre = "Pedro";
280   pedro.apellidos = "Meseguer Morales";
281   pedro.fechaNacimiento = new Fecha();
282   pedro.fechaNacimiento.anyo = 1999;
283   pedro.fechaNacimiento.mes = 8;
284   pedro.fechaNacimiento.dia = 30;
285   pedro.edad = calcularEdad(diferenciaFechas(hoy,
          pedro.fechaNacimiento));
286   pedro.sexo = false;
287   pedro.email = "petermese@um.es";
288   pedro.notaMedia = 0.0;

290   pedro.asignaturas = new Asignatura[3];
291   pedro.asignaturas[0] = new Asignatura();
292   pedro.asignaturas[0].nombre = "Programación";
293   pedro.asignaturas[0].creditos = 6;
294   pedro.asignaturas[0].cuatrimestre = 2;
295   pedro.asignaturas[0].calificacion = random(0, 10);

297   pedro.asignaturas[1] = new Asignatura();
298   pedro.asignaturas[1].nombre = "Álgebra";
299   pedro.asignaturas[1].creditos = 6;
300   pedro.asignaturas[1].cuatrimestre = 1;
301   pedro.asignaturas[1].calificacion = random(0, 10);

303   pedro.asignaturas[2] = new Asignatura();
304   pedro.asignaturas[2].nombre = "Matemática discreta";
305   pedro.asignaturas[2].creditos = 6;
306   pedro.asignaturas[2].cuatrimestre = 2;
307   pedro.asignaturas[2].calificacion = random(0, 10);

309   marina.nombre = "Marina";
310   marina.apellidos = "Antelo Gutiérrez";
311   marina.fechaNacimiento = new Fecha();
312   marina.fechaNacimiento.anyo = 2003;
313   marina.fechaNacimiento.mes = 1;
314   marina.fechaNacimiento.dia = 22;
315   marina.edad = calcularEdad(diferenciaFechas(hoy,
          marina.fechaNacimiento));
316   marina.sexo = true;
317   marina.email = "maguanto@um.es";
318   marina.notaMedia = 0.0;

320   marina.asignaturas = new Asignatura[3];
321   marina.asignaturas[0] = new Asignatura();
322   marina.asignaturas[0].nombre = "Programación";
323   marina.asignaturas[0].creditos = 6;
324   marina.asignaturas[0].cuatrimestre = 2;
325   marina.asignaturas[0].calificacion = random(0, 10);

327   marina.asignaturas[1] = new Asignatura();
```

```
328   marina.asignaturas[1].nombre = "Álgebra";
329   marina.asignaturas[1].creditos = 6;
330   marina.asignaturas[1].cuatrimestre = 1;
331   marina.asignaturas[1].calificacion = random(0, 10);

333   marina.asignaturas[2] = new Asignatura();
334   marina.asignaturas[2].nombre = "Matemática discreta";
335   marina.asignaturas[2].creditos = 6;
336   marina.asignaturas[2].cuatrimestre = 2;
337   marina.asignaturas[2].calificacion = random(0, 10);

339   for (int i = 0; i < clase.alumnos.length; i++) {
340     Alumno alumno = clase.alumnos[i];
341     print("El alumno/a "+alumno.nombre+" "
342       +alumno.apellidos+" nació el "
343       +alumno.fechaNacimiento.dia+"/"
344       +alumno.fechaNacimiento.mes+"/"
345       +alumno.fechaNacimiento.anyo
346       +" (por lo tanto tiene "
347       +alumno.edad.anyos+" años, "
348       +alumno.edad.meses+" meses y "
349       +alumno.edad.dias+" días), su correo electrónico es "
            +alumno.email);
350     if (!alumno.sexo) {
351       println(" y es un hombre");
352     } else {
353       println(" y es una mujer");
354     }

356     for (int j = 0; j < alumno.asignaturas.length; j++) {
357       println("En la asignatura "+alumno.asignaturas[j].nombre+" ("
358         +alumno.asignaturas[j].creditos+" créditos, cuatrimestre "
359         +alumno.asignaturas[j].cuatrimestre+") "+alumno.nombre
360         +" tiene un "+alumno.asignaturas[j].calificacion);
361       alumno.notaMedia += alumno.asignaturas[j].calificacion;
362     }
363     alumno.notaMedia = alumno.notaMedia / alumno.asignaturas.length;
364     println("La nota media de todas las asignaturas de "+alumno.nombre
365       +" es: "+alumno.notaMedia);
366     println();
367   }

369   Alumno mejorAlumno = mejorAlumnoClase(clase);
370   println("Así las cosas, el mejor alumno de clase es "
371   +mejorAlumno.nombre+" con una nota media de "
        +mejorAlumno.notaMedia+"\n");

373   String[] nombresAsignaturas = {"Álgebra", "Matemática discreta", "
        Programación"};
374   for (int i = 0; i < nombresAsignaturas.length; i++) {
375     mejorAlumno = mejorAlumnoAsignatura(clase,
            nombresAsignaturas[i]);
376     println("El mejor alumno de "+nombresAsignaturas[i]+" es "
            +mejorAlumno.nombre);

378     float[] notas = notasAsignatura(clase, nombresAsignaturas[i]);
```

```
379       print("Todas las notas de "+nombresAsignaturas[i]+" son: [");
380       for (int j = 0; j < notas.length-1; j++) {
381         print(notas[j]+", ");
382       }
383       println(notas[notas.length-1]+"]");
384     }
385     println();

387     Edad mediaEdad = mediaEdadClase(clase);
388     println("Por último, la mediad de edad de la clase son "
389     +mediaEdad.anyos+" años, "+mediaEdad.meses+" meses y "
390     +mediaEdad.dias+" días");
391 }
```

6.17 Dada una lista, devolver otra lista con los valores normalizado al rango $[0, 1]$. Por lo tanto, el valor mínimo de la lista devuelta será 0 y el máximo 1.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  float[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }

10   float[] lista = new float[longitud];

12   for (int i = 0; i < longitud; i++) {
13      lista[i] = random(min, max+1);
14   }

16   return lista;
17 }

19 /*
20  * Muestra por pantalla los valores de una
21  * lista dada
22  */
23 void mostrarLista(float[] lista){
24    if (lista != null) {
25      for (int i = 0; i < lista.length; i++){
26        print(lista[i]+" ");
27      }
28    }
29    println();
30 }

32 /*
33 * Esta función devuelve la lista recibida escalada al rango [0, 1]
34 */
35 float[] escalarLista(float[] lista) {
36   float min = lista[0];
37   float max = lista[0];

39   // Buscamos el valor mínimo y máximo
40   for (int i = 1; i < lista.length; i++) {
```

```
    if (lista[i] < min) {
      min = lista[i];
    } else if (lista[i] > max) {
      max = lista[i];
    }
  }
  float[] listaEscalada = new float[lista.length];
  // Normalizamos los valores
  for (int i = 0; i < lista.length; i++) {
    listaEscalada[i] = (lista[i] - min) / (max - min);
  }
  return listaEscalada;
}

void setup(){
  int longitud = 10;
  int min = 5;
  int max = 15;

  float[] lista = crearListaAleatoria(longitud, min, max);
  println("Mi lista aleatoria es:");
  mostrarLista(lista);
  println();

  float[] listaEscalada = escalarLista(lista);
  println("Mi lista escalada es:");
  mostrarLista(listaEscalada);
}
```

6.18 Dada una lista de longitud n con números enteros positivos entre 0 y $n-1$, devolver otra lista que contenga el número de repeticiones de cada número. Para ello, en la posición i-ésima de la lista devuelta debe indicarse el número de repeticiones del número i en la lista original.

```
/*
 * Crea una lista de una longitud dada con valores
 * aleatorios entre un 0 y longitud-1
 */
int[] crearListaAleatoria(int longitud) {
  if (longitud < 1) {
    return null;
  }

  int[] lista = new int[longitud];

  for (int i = 0; i < longitud; i++) {
    lista[i] = int(random(0, longitud));
  }
  return lista;
}

/* Dada una lista de longitud n con números enteros positivos entre 0
    y n-1,
 * devolve otra lista que contiene el número de repeticiones de cada
     número.
 */
int[] contarRepeticiones(int[] lista){
```

```
  // Crea una lista con las repeticiones inicializada a 0
  int longitud = lista.length;
  int[] repeticiones = new int[longitud];
  for (int i = 0; i < longitud; i++) {
    repeticiones[i] = 0;
  }

  // Recorremos la lista y apuntamos las repeticiones
  for (int posicion = 0; posicion < longitud; posicion++) {
    int numero = lista[posicion];
    repeticiones[numero]+= 1;
  }
  return repeticiones;
}


void setup(){
  int longitud = 10;

  // Creamos una lista aleatoria y la mostramos por pantalla
  int[] lista = crearListaAleatoria(longitud);
  println("Mi lista aleatoria es:");
  for (int i = 0; i < lista.length; i++){
    print(lista[i]+" ");
  }
  println();

  // Calculamos las repeticiones
  int[] repeticiones = contarRepeticiones(lista);

  // Mostramos las repeticiones de cada número
  println("Las repeticiones son:");
  for (int i = 0; i < repeticiones.length; i++){
    println("El número "+i+" se repite "+repeticiones[i]);
  }

}
```

6.19 Dado un array de dos dimensiones donde cada celda contiene un número entero, crea un programa que encuentre la fila cuya suma de sus valores sea mayor.

```
/*
 * Crea una matriz de un rango dado con valores
 * aleatorios entre un mínimo y un máximo
 */
int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
    max) {
  if ((filas < 1) || (columnas < 1) || (max < min)) {
    return null;
  }

  int[][] matriz = new int[filas][columnas];

  for (int i = 0; i < filas; i++) {
    for (int j = 0; j < columnas; j++) {
      matriz[i][j] = (int)random(min, max+1);
```

```
15      }
16    }

18    return matriz;
19  }

21  /*
22   * Genera y devuelve una representación de tipo
23   * String de una matriz dada
24   */
25  String matrizAString(int[][] matriz){
26    String cadena = "";

28    if (matriz == null){
29      return cadena;
30    }

32    for (int i = 0; i < matriz.length; i++) {
33      for (int j = 0; j < matriz[0].length; j++) {
34        cadena += matriz[i][j] + "\t";
35      }
36      cadena += "\n";
37    }
38    return cadena;
39  }

41  /*
42   * Muestra por pantalla los valores de una
43   * matriz dada
44   */
45  void mostrarMatriz(int[][] matriz){
46     print(matrizAString(matriz));
47  }

49  /*
50   * Retorna el índice de la fila cuya suma sea mayor
51   */
52  int mayorFila(int[][] matriz) {
53    int maximaSuma = -1;
54    int maximaFila = 0;

56    // Recorremos cada fila de la matriz
57    for (int i = 0; i < matriz.length; i++) {
58      int sumaFila = 0;

60      // Sumamos los números de la fila actual
61      for (int j = 0; j < matriz[i].length; j++) {
62        sumaFila += matriz[i][j];
63      }

65      // Comparamos la suma de la fila actual
66      // con la suma máxima encontrada hasta ahora
67      if (sumaFila > maximaSuma) {
68        maximaSuma = sumaFila;
69        maximaFila = i;
70      }
```

```
71    }
72    return maximaFila + 1;
73  }

75  void setup(){
76    int filas = 3;
77    int columnas = 3;
78    int min = 5;
79    int max = 15;

81    int[][] matriz = crearMatrizAleatoria(filas, columnas, min, max);
82    println("La matriz aleatoria es: ");
83    mostrarMatriz(matriz);
84    int maximaFila = mayorFila(matriz);
85    println("La fila con la mayor suma es "+ maximaFila);
86  }
```

6.20 Calcula el determinante de una matriz de 3×3.

```
1   /*
2    * Crea una matriz de un rango dado con valores
3    * aleatorios entre un mínimo y un máximo
4    */
5   int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
        max) {
6     if ((filas < 1) || (columnas < 1) || (max < min)) {
7       return null;
8     }

10    int[][] matriz = new int[filas][columnas];

12    for (int i = 0; i < filas; i++) {
13      for (int j = 0; j < columnas; j++) {
14        matriz[i][j] = (int)random(min, max+1);
15      }
16    }

18    return matriz;
19  }



22  /*
23   * Genera y devuelve una representación de tipo
24   * String de una matriz dada.
25   */
26  String matrizAString(int[][] matriz){
27    String cadena = "";

29    if (matriz == null){
30      return cadena;
31    }

33    for (int i = 0; i < matriz.length; i++) {
34      for (int j = 0; j < matriz[0].length; j++) {
35        cadena += matriz[i][j] + "\t";
36      }
37      cadena += "\n";
```

```
38    }
39    return cadena;
40  }

42  /*
43   * Muestra por pantalla los valores de una
44   * matriz dada
45   */
46  void mostrarMatriz(int[][] matriz){
47     print(matrizAString(matriz));
48  }

50  /*
51   * Calcula el determiante de una matriz 3x3
52   */
53  int calcularDeterminante(int matriz [][]) {

55    // Si la matriz no es 3x3
56    if((matriz.length != 3) || (matriz[0].length != 3)){
57      println("La matriz debe ser 3x3 dimensiones");
58      return 0;
59    }

61    int dimensiones = matriz.length;
62    int determinante = 0;

64    // Recorremos cada columna
65    for (int columna = 0; columna < dimensiones; columna++){

67      int determinanteDiagonalSumar = matriz[0][columna];
68      int determinanteDiagonalRestar = matriz[0][columna];

70      // Desplazamos por las filas y columnas
71      for (int desplazamiento = 1; desplazamiento < dimensiones;
            desplazamiento++){
72        // Desplazamiento hacia abajo y derecha
73        determinanteDiagonalSumar = determinanteDiagonalSumar *
              matriz[desplazamiento][(columna+desplazamiento)%dimensiones];
74        // Desplazamiento hacia abajo e izquierda
75        determinanteDiagonalRestar = determinanteDiagonalRestar *
              matriz[desplazamiento][(columna-desplazamiento +
              dimensiones)%dimensiones];
76      }
77      determinante = determinante + determinanteDiagonalSumar -
            determinanteDiagonalRestar;
78    }
79    return determinante;
80  }

82  void setup(){
83    int filas = 3;
84    int columnas = 3;
85    int min = -5;
86    int max = 15;
```

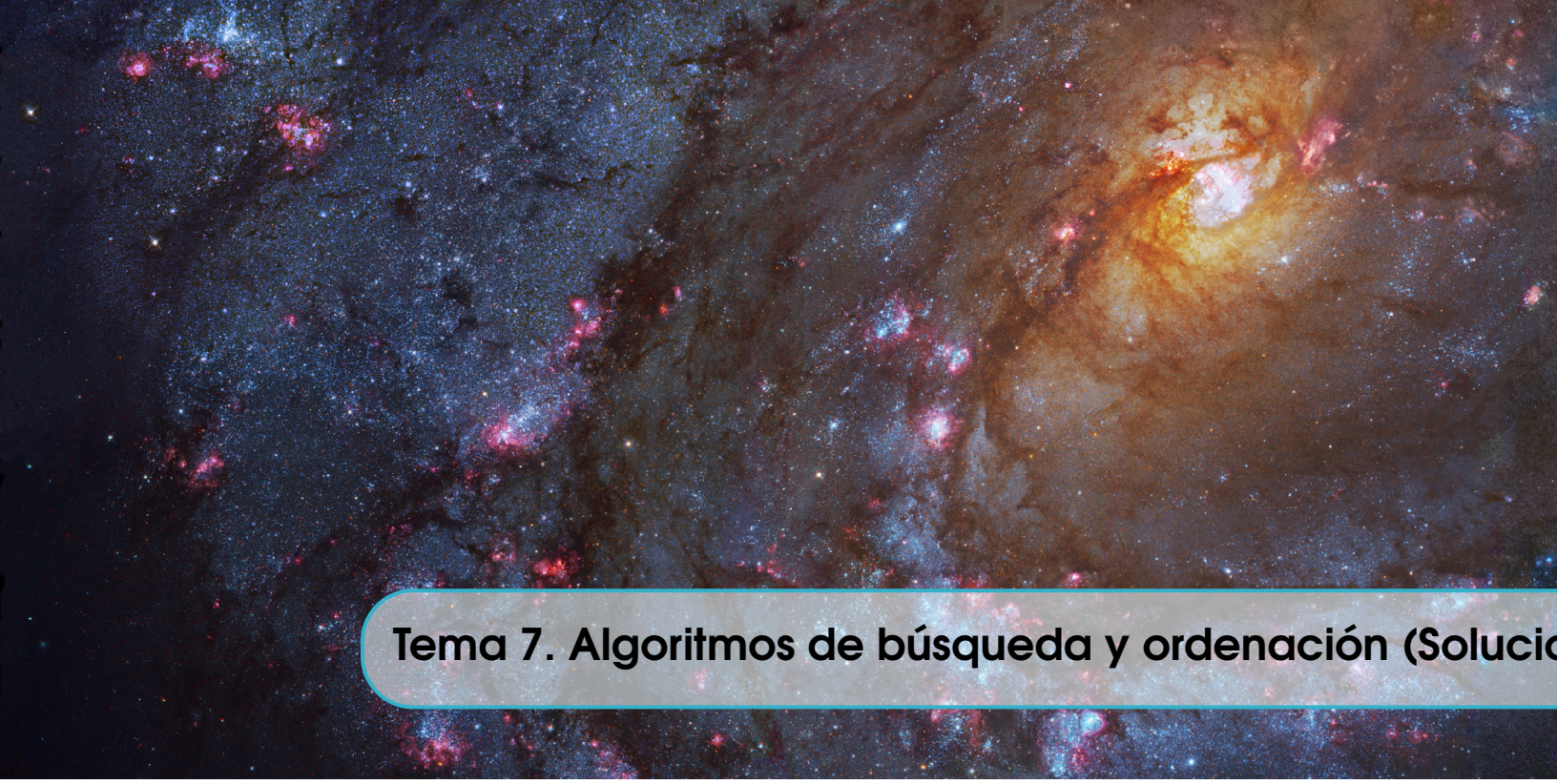
```
89    int[][] miMatriz = crearMatrizAleatoria(filas, columnas, min, max);
90    println("Mi matriz aleatoria es:");
91    mostrarMatriz(miMatriz);
92    println();
93    int determinante = calcularDeterminante(miMatriz);
94    println("Tiene el determinante: "+str(determinante));


97  }
```

6.21 Vamos a representar el mapa de un terreno mediante un array de dos dimensiones, siendo valor de cada casilla la elevación del terreno en esas coordenadas. Se desea realizar una operación de suavizado para obtener una versión más suave del terreno. Esto ayuda a eliminar las irregularidades en la elevación del terreno y a obtener una versión uniforme del mismo. Una forma de suavizar el terreno es promediar el valor de cada punto con el de sus ocho vecinos inmediatos, que representan los puntos en las direcciones horizontal, vertical y diagonal. El resultado de la operación de suavizado es un nuevo array de dos dimensiones donde cada casilla contenga el promedio de la casilla original y de sus vecinos que estén a una casilla de distancia. Ten en cuenta que las casillas situadas en los bordes tienen menos vecinos con los que promediar.

```
1   /*
2    * Crea una matriz de un rango dado con valores
3    * aleatorios entre un mínimo y un máximo
4    */
5   float[][] crearMatrizAleatoria(int filas, int columnas, int min, int
        max) {
6     if ((filas < 1) || (columnas < 1) || (max < min)) {
7       return null;
8     }

10    float[][] matriz = new float[filas][columnas];

12    for (int i = 0; i < filas; i++) {
13      for (int j = 0; j < columnas; j++) {
14        matriz[i][j] = random(min, max+1);
15      }
16    }

18    return matriz;
19  }


22  /*
23   * Genera y devuelve una representación de tipo
24   * String de una matriz dada.
25   */
26  String matrizAString(float[][] matriz){
27    String cadena = "";

29    if (matriz == null){
30      return cadena;
31    }

33    for (int i = 0; i < matriz.length; i++) {
```

```
34      for (int j = 0; j < matriz[0].length; j++) {
35        cadena += matriz[i][j]  + " ";
36      }
37      cadena += "\n";
38    }
39    return cadena;
40  }

42  /*
43   * Muestra por pantalla los valores de una
44   * matriz dada
45   */
46  void mostrarMatriz(float[][] matriz){
47     print(matrizAString(matriz));
48  }

50  /*
51   * Retorna una matriz donde cada casilla contenga el promedio de la
         casilla original
52   * y de sus vecinos que estén a una casilla de distancia.
53   */

55  float[][] suavizarTerreno(float[][] terreno) {
56    int filas = terreno.length;
57    int columnas = terreno[0].length;
58    float[][] terrenoSuavizado = new float[filas][columnas];

60    // Recorremos todas las posiciones del terreno
61    for (int i = 0; i < filas; i++) {
62      for (int j = 0; j < columnas; j++) {
63        float suma = 0;
64        int contador = 0;

66        // Calculamos la suma y el número de vecinos para el punto actual
67        for (int k = -1; k <= 1; k++) { // moviento en las filas contiguas
68          for (int l = -1; l <= 1; l++) { // moviento en las columnas
                contiguas
69            if ((i + k >= 0) && (i + k < filas) && (j + l >= 0) && (j +
                 l < columnas)) {
70              suma += terreno[i + k][j + l];
71              contador++;
72            }
73          }
74        }

76        // Calculamos el valor promedio para el punto actual
77        terrenoSuavizado[i][j] = suma / contador;
78      }
79    }

81    return terrenoSuavizado;
82  }

84  void setup(){
85    int filas = 4;
86    int columnas = 5;
87    int min = 5;
```

```
88    int max = 15;

90    float[][] terreno = crearMatrizAleatoria(filas, columnas, min, max);
91    println("El terreno original es:");
92    mostrarMatriz(terreno);
93    println();

95    float[][] terrenoSuavizado = suavizarTerreno(terreno);
96    println("El terreno suavizado es:");
97    mostrarMatriz(terrenoSuavizado);
98  }
```



Tema 7. Algoritmos de búsqueda y ordenación (Soluci

7.1 Tema 7: Ejercicios para clase de teoría

7.1 Sobre una lista de naturales (enteros positivos) construir las siguientes funciones:

- Indicar si un número dado se encuentra o no contenido en la lista.
- Contar el número de veces que un número dado se encuentra en la lista.
- Devolver las posiciones que ocupa un número dado en la lista.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }
9
10   int[] lista = new int[longitud];
11
12   for (int i = 0; i < longitud; i++) {
13      lista[i] = (int)random(min, max+1);
14   }
15
16   return lista;
17 }
18
19 /*
20  * Busca un valor dado en una lista dada.
21  * Devuelve true si encuentra el valor y
22  * false si no lo encuentra
23  */
24 boolean buscarValor(int[] lista, int valor){
25    if (lista != null) {
26      for (int i = 0; i < lista.length; i++){
27         if (lista[i] == valor) {
```

```
28              return true;
29           }
30        }
31     }
32     return false;
33  }

35  /*
36   * Cuenta el número de ocurrencias de un valor dado
37   * en una lista de números dada.
38   */
39  int contarOcurrencias(int[] lista, int valor) {
40     int contador = 0;
41     if (lista != null) {
42        for (int i = 0; i < lista.length; i++){
43           if (lista[i] == valor) {
44              contador++;
45           }
46        }
47     }
48     return contador;
49  }

51  /*
52   * Devuelve las posiciones que ocupa un valor dado
53   * en una lista de números dada.
54   */
55  int[] posicionesValor(int[] lista, int valor) {
56     int contador = contarOcurrencias(lista, valor);
57     int[] posiciones = new int[contador];
58     contador = 0;
59     if (lista != null) {
60        for (int i = 0; i < lista.length; i++){
61           if (lista[i] == valor) {
62              posiciones[contador] = i;
63              contador++;
64           }
65        }
66     }
67     return posiciones;
68  }


71  /*
72   * Muestra por pantalla los valores de una
73   * lista dada
74   */
75  void mostrarLista(int[] lista){
76     if (lista != null) {
77        for (int i = 0; i < lista.length; i++){
78          print(lista[i]+" ");
79        }
80     }
81     println();
82  }
```

```
84 void setup(){
85   int longitud = 10;
86   int min = 5;
87   int max = 15;
88   int N = 10;

90   int[] miLista = crearListaAleatoria(longitud, min, max);
91   println("Mi lista aleatoria de longitud "+longitud+" y valores
         entre "+min+" y "+max+" es:");
92   mostrarLista(miLista);
93   println();

95   for (int i = 0; i < N; i++) {
96      int numero = (int)random(min, max);
97      if (buscarValor(miLista, numero)) {
98        int contador = contarOcurrencias(miLista, numero);
99        int[] posiciones = posicionesValor(miLista, numero);
100       println("El número "+numero+" aparece "+contador+" veces en la
              lista");
101       print("Y ocupa las posiciones: ");
102       mostrarLista(posiciones);
103       println("\n");
104     } else {
105       println("El número "+numero+" no aparece en la lista\n");
106     }
107  }

109 }
```

7.2 Sobre una matriz de números naturales, construye cada una de las funciones cuyo propósito se indica a continuación:

- Indicar si un número dado se encuentra o no contenido en la matriz.
- Contar el número de veces que un número dado se encuentra en la matriz.

```
1  /*
2   * Crea una matriz de un rango dado con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
       max) {
6    if ((filas < 1) || (columnas < 1) || (max < min)) {
7      return null;
8    }

10   int[][] matriz = new int[filas][columnas];

12   for (int i = 0; i < filas; i++) {
13     for (int j = 0; j < columnas; j++) {
14       matriz[i][j] = (int)random(min, max+1);
15     }
16   }

18   return matriz;
19 }

21 /*
```

```
22  * Busca un valor dado en una matriz dada.
23  * Devuelve true si encuentra el valor y
24  * false si no lo encuentra.
25  */
26 boolean buscarValor(int[][] matriz, int valor){
27   if (matriz == null) {
28     return false;
29   }
30   for (int i = 0; i < matriz.length; i++) {
31     for (int j = 0; j < matriz[i].length; j++) {
32       if (matriz[i][j] == valor) {
33         return true;
34       }
35     }
36   }
37   return false;
38 }

40 /*
41  * Cuenta el número de ocurrencias de un valor
42  * dado en una matriz dada.
43  */
44 int contarOcurrencias(int[][] matriz, int valor){
45   int contador = 0;
46   if (matriz != null) {
47     for (int i = 0; i < matriz.length; i++) {
48       for (int j = 0; j < matriz[i].length; j++) {
49         if (matriz[i][j] == valor) {
50           contador++;
51         }
52       }
53     }
54   }
55   return contador;
56 }

58 /*
59  * Genera y devuelve una representación de tipo
60  * String de una matriz dada.
61  */
62 String matrizAString(int[][] matriz){
63   String cadena = "";

65   if (matriz == null){
66     return cadena;
67   }

69   for (int i = 0; i < matriz.length; i++) {
70     for (int j = 0; j < matriz[0].length; j++) {
71       cadena += matriz[i][j] + "\t";
72     }
73     cadena += "\n";
74   }
75   return cadena;
76 }
```

```
78 /*
79  * Muestra por pantalla los valores de una
80  * matriz dada
81  */
82 void mostrarMatriz(int[][] matriz){
83    print(matrizAString(matriz));
84 }

86 void setup(){
87   int filas = 4;
88   int columnas = 5;
89   int min = 5;
90   int max = 15;
91   int N = 10;

93   int[][] miMatriz = crearMatrizAleatoria(filas, columnas, min, max);
94   println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
         columnas, con valores entre "+min+" y "+max+" es:");
95   mostrarMatriz(miMatriz);
96   println();

98   for (int i = 0; i < N; i++) {
99     int numero = (int)random(min, max);

101    if (buscarValor(miMatriz, numero)) {
102      int contador = contarOcurrencias(miMatriz, numero);
103      println("El número "+numero+" aparece "+contador+" veces en la
           matriz\n");
104    } else {
105      println("El número "+numero+" no aparece en la matriz\n");
106    }
107  }
108 }
```

7.3 Crea un registro para representar a una posición (i, j) dentro de una matriz. Amplía el ejercicio **7.2** para devolver todas las posiciones que ocupa un número dado en una matriz dada.

```
1  class Posicion{
2    int fila;
3    int columna;
4  }

6  /*
7   * Crea una matriz de un rango dado con valores
8   * aleatorios entre un mínimo y un máximo
9   */
10 int[][] crearMatrizAleatoria(int filas, int columnas, int min, int
       max) {
11   if ((filas < 1) || (columnas < 1) || (max < min)) {
12     return null;
13   }

15   int[][] matriz = new int[filas][columnas];

17   for (int i = 0; i < filas; i++) {
18     for (int j = 0; j < columnas; j++) {
19       matriz[i][j] = (int)random(min, max+1);
```

```
20      }
21    }

23    return matriz;
24  }

26  /*
27   * Busca un valor dado en una matriz dada.
28   * Devuelve true si encuentra el valor y
29   * false si no lo encuentra.
30   */
31  boolean buscarValor(int[][] matriz, int valor){
32    if (matriz == null) {
33      return false;
34    }
35    for (int i = 0; i < matriz.length; i++) {
36      for (int j = 0; j < matriz[i].length; j++) {
37        if (matriz[i][j] == valor) {
38          return true;
39        }
40      }
41    }
42    return false;
43  }

45  /*
46   * Cuenta el número de ocurrencias de un valor
47   * dado en una matriz dada.
48   */
49  int contarOcurrencias(int[][] matriz, int valor){
50    int contador = 0;
51    if (matriz != null) {
52      for (int i = 0; i < matriz.length; i++) {
53        for (int j = 0; j < matriz[i].length; j++) {
54          if (matriz[i][j] == valor) {
55            contador++;
56          }
57        }
58      }
59    }
60    return contador;
61  }

63  /*
64   * Devuelve las posiciones que ocupa un valor dado
65   * en una matriz dada.
66   */
67  Posicion[] posicionesValor(int[][]matriz, int valor){
68    int contador = contarOcurrencias(matriz, valor);
69    Posicion[] posiciones = new Posicion[contador];
70    contador = 0;
71    if (matriz != null) {
72      for (int i = 0; i < matriz.length; i++) {
73        for (int j = 0; j < matriz[i].length; j++) {
74          if (matriz[i][j] == valor) {
75            posiciones[contador] = new Posicion();
```

```
76            posiciones[contador].fila = i;
77            posiciones[contador].columna = j;
78            contador++;
79          }
80        }
81      }
82    }
83    return posiciones;
84  }

86  /*
87   * Genera y devuelve una representación de tipo
88   * String de una matriz dada.
89   */
90  String matrizAString(int[][] matriz){
91    String cadena = "";

93    if (matriz == null){
94      return cadena;
95    }

97    for (int i = 0; i < matriz.length; i++) {
98      for (int j = 0; j < matriz[0].length; j++) {
99        cadena += matriz[i][j] + "\t";
100     }
101     cadena += "\n";
102   }
103   return cadena;
104 }

106 /*
107  * Muestra por pantalla los valores de una
108  * matriz dada
109  */
110 void mostrarMatriz(int[][] matriz){
111    print(matrizAString(matriz));
112 }

114 void setup(){
115   int filas = 4;
116   int columnas = 5;
117   int min = 5;
118   int max = 15;
119   int N = 10;

121   int[][] miMatriz = crearMatrizAleatoria(filas, columnas, min, max);
122   println("Mi matriz aleatoria A con "+filas+" filas y "+columnas+"
        columnas, con valores entre "+min+" y "+max+" es:");
123   mostrarMatriz(miMatriz);
124   println();

126   for (int i = 0; i < N; i++) {
127     int numero = (int)random(min, max);

129     if (buscarValor(miMatriz, numero)) {
130       int contador = contarOcurrencias(miMatriz, numero);
```

```
131       Posicion[] posiciones = posicionesValor(miMatriz, numero);
132       println("El número "+numero+" aparece "+contador+" veces en la
              matriz\n");
133       print("Y ocupa las posiciones: ");
134       for (int j = 0; j < posiciones.length; j++) {
135         print("("+posiciones[j].fila+", "+posiciones[j].columna+") ");
136       }
137       println("\n");
138     } else {
139       println("El número "+numero+" no aparece en la matriz\n");
140     }
141   }
142 }
```

7.2 Tema 7: Ejercicios para clase de prácticas

7.4 Dado un array de enteros, implementar las siguientes funciones:

- Ordenar la lista de forma ascendente o descendente, según un parámetro de la función.
- Devolver otro array eliminando los números repetidos (independientemente de que el array esté ordenado o no). Por ejemplo, dado el array $\begin{bmatrix}4 & 2 & 2 & 4 & 5 & 6\end{bmatrix}$ debería devolver $\begin{bmatrix}4 & 2 & 5 & 6\end{bmatrix}$.
- Devolver otro array eliminando los números repetidos y que además sus elementos estén ordenados de forma ascendente o descendente, según un parámetro de la función.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }
9
10   int[] lista = new int[longitud];
11
12   for (int i = 0; i < longitud; i++) {
13      lista[i] = (int)random(min, max+1);
14   }
15
16   return lista;
17 }
18
19 /*
20  * Ordena una lista dada de forma ascendente o
21  * descendente (dependiendo del parámetro ascendente)
22  * utilizando el método burbuja.
23  */
24 void ordenarLista(int[] lista, boolean ascendente) {
25    int longitud = lista.length;
26
27    do {
28      int nuevaLongitud = 0;
29      for (int i = 1; i <= longitud-1; i++) {
30        if ((ascendente && (lista[i-1] > lista[i])) ||
31            (!ascendente && (lista[i-1] < lista[i]))) {
32          int aux = lista[i-1];
33          lista[i-1] = lista[i];
34          lista[i] = aux;
35          nuevaLongitud = i;
36        }
37      }
38      longitud = nuevaLongitud;
39    } while(longitud != 0);
40 }
41
42 /*
43  * Devuelve una copia de una lista dada.
44  */
45 int[] copiarLista(int[] lista) {
46   if (lista == null) {
```

```
47      return null;
48    }

50    int[] copia = new int[lista.length];
51    for (int i = 0; i < lista.length; i++) {
52      copia[i] = lista[i];
53    }

55    return copia;
56  }

58  /*
59   * Devuelve una nueva lista, eliminando el elemento que
60   * ocupa una posición dada.
61   */
62  int[] eliminarElementoLista(int[] lista, int posicion) {
63    if ((lista == null) || (posicion < 0) || (posicion >=
          lista.length)) {
64      return lista;
65    }

67    int[] nuevaLista = new int[lista.length-1];
68    for (int i = 0, j = 0; i < lista.length; i++) {
69      if (i != posicion) {
70        nuevaLista[j++] = lista[i];
71      }
72    }
73    return nuevaLista;
74  }

76  /*
77   * Devuelve una nueva lista en la que se eliminan todos los
78   * elementos repetidos de una lista dada, dejando solamente
79   * una ocurrencia de los elementos que estén repetidos.
80   */
81  int[] eliminarRepetidos(int[] lista) {
82    if ((lista == null) || (lista.length < 2)) {
83      return lista;
84    }

86    int[] copia = copiarLista(lista);
87    for (int i = 1; i < copia.length; i++) {
88      //Recorremos desde la posición i-1 hacia el principio del
89      //array, y si encontramos un elemento igual al elemento
90      //de la posición i-ésima, eliminamos ese elemento
91      for (int j = i-1; j >= 0; j--) {
92        if (copia[i] == copia[j]) {
93          copia = eliminarElementoLista(copia, i);
94          i--;
95        }
96      }
97    }

99    return copia;
100 }
```

```
/*
 * Devuelve una nueva lista, con o sin elementos repetidos
 * (dependiendo del parámetro sinRepetidos) y ordenada de
 * forma ascendente o descendente (dependiendo del parámetro
 * ascendiente).
 */
int[] ordenarLista(int[] lista, boolean ascendente, boolean
    sinRepetidos) {
  int[] copia = copiarLista(lista);
  if (sinRepetidos) {
     copia = eliminarRepetidos(copia);
  }
  ordenarLista(copia, ascendente);
  return copia;
}

/*
 * Muestra por pantalla los valores de una
 * lista dada
 */
void mostrarLista(int[] lista){
   if (lista != null) {
     for (int i = 0; i < lista.length; i++){
       print(lista[i]+" ");
     }
   }
   println();
}

void setup(){
  int longitud = 10;
  int min = 5;
  int max = 15;
  int N = 10;

  for (int i = 0; i < N; i++) {
    int[] miLista = crearListaAleatoria(longitud, min, max);
    println("Mi lista aleatoria de longitud "+longitud+" y valores
        entre "+min+" y "+max+" es:");
    mostrarLista(miLista);

    boolean ascendente = true;
    if (random(0,1) > 0.5) {
      ascendente = false;
    }
    boolean sinRepetidos = true;
    if (random(0,1) > 0.5) {
      sinRepetidos = false;
    }

    miLista = ordenarLista(miLista, ascendente, sinRepetidos);
    if (ascendente) {
      print("Después de ordenarla de forma ascendente, ");
    } else {
      print("Después de ordenarla de forma descendente, ");
    }
```

```
156     if (sinRepetidos) {
157       println("y eliminando los números repetidos, la lista queda
              así:");
158     } else {
159       println("y sin eliminar los números repetidos, la lista queda
              así:");
160     }
161     mostrarLista(miLista);
162     println();
163   }
164 }
```

7.5 A partir del ejercicio **6.9**, ordenar en orden ascendente las asignaturas de cada alumno en función de su calificación.

7.3 Tema 7: Ejercicios para casa

7.6 A partir del ejercicio **6.16**, crea las siguientes funciones:

- Ordenar todos los alumnos de una clase en función de su nota media.
- Ordenar todos los alumnos de una clase en función de la nota de una asignatura dada.

```
1  class Clase {
2    int curso;
3    Alumno[] alumnos;
4  }

6  class Fecha {
7    int dia;
8    int mes;
9    int anyo;
10 }

12 class Edad {
13   int anyos;
14   int meses;
15   int dias;
16 }

18 class Asignatura {
19   String nombre;
20   int creditos;
21   int cuatrimestre;
22   float calificacion;
23 }

25 class Alumno {
26   String nombre;
27   String apellidos;
28   Fecha fechaNacimiento;
29   Edad edad;
30   boolean sexo; //True para mujer y false para hombre
31   String email;

33   Asignatura[] asignaturas;
34   float notaMedia;
35 }

37 /*
38  * Calcula la diferencia entre dos fechas dadas
39  * fecha1 y fecha2, expresada en días.
40  * Si fecha2 es posterior a fecha1, entonces
41  * devuelve -1 como código de error.
42  */
43 int diferenciaFechas(Fecha fecha1, Fecha fecha2) {
44   int numDiasFecha1 = fecha1.anyo*365 + (fecha1.mes-1)*30 +
       fecha1.dia;
45   int numDiasFecha2 = fecha2.anyo*365 + (fecha2.mes-1)*30 +
       fecha2.dia;

47   if (numDiasFecha2 > numDiasFecha1) {
48     return -1;
49   }
```

```
51    return numDiasFecha1 - numDiasFecha2;
52  }
53
54  /*
55   * Calcula una edad (años, meses y días)
56   * a partir de un número total de días
57   */
58  Edad calcularEdad(int numeroDias) {
59    Edad edad = new Edad();
60
61    edad.anyos = numeroDias / 365;
62    edad.meses = (numeroDias % 365) / 30;
63    edad.dias = (numeroDias % 365) % 30;
64
65    return edad;
66  }
67
68  /*
69   * Convierte una edad en su equivalente
70   * en días.
71   */
72  int edadADias(Edad edad) {
73     return edad.anyos*365 + (edad.meses-1)*30 + edad.dias;
74  }
75
76  /*
77   * Devuelve el alumno con la máxima nota
78   * media de una clase dada.
79   */
80  Alumno mejorAlumnoClase(Clase clase) {
81    float mejorNota = -1;
82    Alumno mejorAlumno = null;
83
84    for (int i = 0; i < clase.alumnos.length; i++) {
85      Alumno alumno = clase.alumnos[i];
86      if (alumno.notaMedia > mejorNota) {
87        mejorNota = alumno.notaMedia;
88        mejorAlumno = alumno;
89      }
90    }
91    return mejorAlumno;
92  }
93
94  /*
95   * Devuelve el alumno con la máxima nota
96   * media de una clase dada, para una
97   * asignatura dada.
98   */
99  Alumno mejorAlumnoAsignatura(Clase clase, String nombreAsignatura) {
100   float mejorNota = -1;
101   Alumno mejorAlumno = null;
102
103   for (int i = 0; i < clase.alumnos.length; i++) {
104     Alumno alumno = clase.alumnos[i];
105     for (int j = 0; j < alumno.asignaturas.length; j++) {
```

```
106         Asignatura asignatura = alumno.asignaturas[j];
107         if ((asignatura.nombre.equals(nombreAsignatura))
108          &&( asignatura.calificacion > mejorNota)) {
109           mejorNota = asignatura.calificacion;
110           mejorAlumno = alumno;
111         }
112       }
113     }
114     return mejorAlumno;
115 }

117 /*
118  * Devuelve un array con las notas de una lista de
119  * alumnos dada, para una asignatura dada.
120  */
121 float[] notasAsignatura(Alumno[] alumnos, String nombreAsignatura) {
122     int numNotas = 0;

124     for (int i = 0; i < alumnos.length; i++) {
125       Alumno alumno = alumnos[i];
126       for (int j = 0; j < alumno.asignaturas.length; j++) {
127         Asignatura asignatura = alumno.asignaturas[j];
128         if (asignatura.nombre.equals(nombreAsignatura)) {
129           numNotas++;
130         }
131       }
132     }

134     float[] notas = new float[numNotas];
135     numNotas = 0;

137     for (int i = 0; i < alumnos.length; i++) {
138       Alumno alumno = alumnos[i];
139       for (int j = 0; j < alumno.asignaturas.length; j++) {
140         Asignatura asignatura = alumno.asignaturas[j];
141         if (asignatura.nombre.equals(nombreAsignatura)) {
142           notas[numNotas] = asignatura.calificacion;
143           numNotas++;
144         }
145       }
146     }

148     return notas;
149 }

151 /*
152  * Devuelve un array con las notas de los
153  * alumnos de una clase dada, para una
154  * asignatura dada.
155  */
156 float[] notasAsignatura(Clase clase, String nombreAsignatura) {
157     return notasAsignatura(clase.alumnos, nombreAsignatura);
158 }

160 /*
161  * Calcula la edad media de los
```

```
 * alumnos de una clase dada.
 */
Edad mediaEdadClase(Clase clase) {
  int mediaEdadEnDias = 0;
  for (int i = 0; i < clase.alumnos.length; i++) {
    Alumno alumno = clase.alumnos[i];
    mediaEdadEnDias += edadADias(alumno.edad);
  }
  mediaEdadEnDias = mediaEdadEnDias / clase.alumnos.length;

  return calcularEdad(mediaEdadEnDias);
}

/*
 * Ordena una lista dada de alumnos a partir de sus
 * notas medias de forma ascendente o descendente
 * (dependiendo del parámetro ascendente) utilizando el
 * método burbuja.
 */
void ordenarAlumnosNotaMedia(Alumno[] alumnos, boolean ascendente) {
   int longitud = alumnos.length;

   do {
     int nuevaLongitud = 0;
     for (int i = 1; i <= longitud-1; i++) {
       if ((ascendente && (alumnos[i-1].notaMedia >
           alumnos[i].notaMedia)) ||
           (!ascendente && (alumnos[i-1].notaMedia <
               alumnos[i].notaMedia))) {
          Alumno aux = alumnos[i-1];
          alumnos[i-1] = alumnos[i];
          alumnos[i] = aux;
          nuevaLongitud = i;
       }
     }
     longitud = nuevaLongitud;
   } while(longitud != 0);
}

/*
 * Ordena una lista dada de alumnos a partir de la califiación
 * de una asignatura dada, de forma ascendente o descendente
 * (dependiendo del parámetro ascendente) utilizando el
 * método burbuja.
 */
void ordenarAlumnosPorAsignatura(Alumno[] alumnos, boolean
    ascendente, String nombreAsignatura) {
   int longitud = alumnos.length;
   float[] calificaciones = notasAsignatura(alumnos,
       nombreAsignatura);
   do {
     int nuevaLongitud = 0;
     for (int i = 1; i <= longitud-1; i++) {
       if ((ascendente && (calificaciones[i-1] > calificaciones[i]))
           ||
           (!ascendente && (calificaciones[i-1] <
```

```
                calificaciones[i]))) {
213           Alumno aux = alumnos[i-1];
214           alumnos[i-1] = alumnos[i];
215           alumnos[i] = aux;
216           float aux2 = calificaciones[i-1];
217           calificaciones[i-1] = calificaciones[i];
218           calificaciones[i] = aux2;
219           nuevaLongitud = i;
220         }
221       }
222       longitud = nuevaLongitud;
223     } while(longitud != 0);
224 }

226 void setup() {
227   Clase clase = new Clase();
228   clase.curso = 1;
229   clase.alumnos = new Alumno[5];

231   Fecha hoy = new Fecha();
232   hoy.anyo = 2022;
233   hoy.mes = 3;
234   hoy.dia = 23;

236   Alumno alberto = new Alumno();
237   Alumno manuel = new Alumno();
238   Alumno sofia = new Alumno();
239   Alumno pedro = new Alumno();
240   Alumno marina = new Alumno();

242   clase.alumnos[0] = alberto;
243   clase.alumnos[1] = manuel;
244   clase.alumnos[2] = sofia;
245   clase.alumnos[3] = pedro;
246   clase.alumnos[4] = marina;

248   alberto.nombre = "Alberto";
249   alberto.apellidos = "García Martínez";
250   alberto.fechaNacimiento = new Fecha();
251   alberto.fechaNacimiento.anyo = 2001;
252   alberto.fechaNacimiento.mes = 7;
253   alberto.fechaNacimiento.dia = 20;
254   alberto.edad = calcularEdad(diferenciaFechas(hoy,
        alberto.fechaNacimiento));
255   alberto.sexo = false;
256   alberto.email = "alberto.garcia-martinez@um.es";
257   alberto.notaMedia = 0.0;

259   alberto.asignaturas = new Asignatura[3];
260   alberto.asignaturas[0] = new Asignatura();
261   alberto.asignaturas[0].nombre = "Programación";
262   alberto.asignaturas[0].creditos = 6;
263   alberto.asignaturas[0].cuatrimestre = 2;
264   alberto.asignaturas[0].calificacion = random(0, 10);

266   alberto.asignaturas[1] = new Asignatura();
```

```
267   alberto.asignaturas[1].nombre = "Álgebra";
268   alberto.asignaturas[1].creditos = 6;
269   alberto.asignaturas[1].cuatrimestre = 1;
270   alberto.asignaturas[1].calificacion = random(0, 10);

272   alberto.asignaturas[2] = new Asignatura();
273   alberto.asignaturas[2].nombre = "Matemática discreta";
274   alberto.asignaturas[2].creditos = 6;
275   alberto.asignaturas[2].cuatrimestre = 2;
276   alberto.asignaturas[2].calificacion = random(0, 10);

278   manuel.nombre = "Manuel";
279   manuel.apellidos = "Buitrago Gil";
280   manuel.fechaNacimiento = new Fecha();
281   manuel.fechaNacimiento.anyo = 2001;
282   manuel.fechaNacimiento.mes = 12;
283   manuel.fechaNacimiento.dia = 2;
284   manuel.edad = calcularEdad(diferenciaFechas(hoy,
          manuel.fechaNacimiento));
285   manuel.sexo = false;
286   manuel.email = "manuel.bg@um.es";
287   manuel.notaMedia = 0.0;

289   manuel.asignaturas = new Asignatura[3];
290   manuel.asignaturas[0] = new Asignatura();
291   manuel.asignaturas[0].nombre = "Programación";
292   manuel.asignaturas[0].creditos = 6;
293   manuel.asignaturas[0].cuatrimestre = 2;
294   manuel.asignaturas[0].calificacion = random(0, 10);

296   manuel.asignaturas[1] = new Asignatura();
297   manuel.asignaturas[1].nombre = "Álgebra";
298   manuel.asignaturas[1].creditos = 6;
299   manuel.asignaturas[1].cuatrimestre = 1;
300   manuel.asignaturas[1].calificacion = random(0, 10);

302   manuel.asignaturas[2] = new Asignatura();
303   manuel.asignaturas[2].nombre = "Matemática discreta";
304   manuel.asignaturas[2].creditos = 6;
305   manuel.asignaturas[2].cuatrimestre = 2;
306   manuel.asignaturas[2].calificacion = random(0, 10);

308   sofia.nombre = "Sofía";
309   sofia.apellidos = "López Pérez";
310   sofia.fechaNacimiento = new Fecha();
311   sofia.fechaNacimiento.anyo = 2002;
312   sofia.fechaNacimiento.mes = 5;
313   sofia.fechaNacimiento.dia = 18;
314   sofia.edad = calcularEdad(diferenciaFechas(hoy,
          sofia.fechaNacimiento));
315   sofia.sexo = true;
316   sofia.email = "sofialp@um.es";
317   sofia.notaMedia = 0.0;

319   sofia.asignaturas = new Asignatura[3];
320   sofia.asignaturas[0] = new Asignatura();
```

```
321   sofia.asignaturas[0].nombre = "Programación";
322   sofia.asignaturas[0].creditos = 6;
323   sofia.asignaturas[0].cuatrimestre = 2;
324   sofia.asignaturas[0].calificacion = random(0, 10);

326   sofia.asignaturas[1] = new Asignatura();
327   sofia.asignaturas[1].nombre = "Álgebra";
328   sofia.asignaturas[1].creditos = 6;
329   sofia.asignaturas[1].cuatrimestre = 1;
330   sofia.asignaturas[1].calificacion = random(0, 10);

332   sofia.asignaturas[2] = new Asignatura();
333   sofia.asignaturas[2].nombre = "Matemática discreta";
334   sofia.asignaturas[2].creditos = 6;
335   sofia.asignaturas[2].cuatrimestre = 2;
336   sofia.asignaturas[2].calificacion = random(0, 10);

338   pedro.nombre = "Pedro";
339   pedro.apellidos = "Meseguer Morales";
340   pedro.fechaNacimiento = new Fecha();
341   pedro.fechaNacimiento.anyo = 1999;
342   pedro.fechaNacimiento.mes = 8;
343   pedro.fechaNacimiento.dia = 30;
344   pedro.edad = calcularEdad(diferenciaFechas(hoy,
          pedro.fechaNacimiento));
345   pedro.sexo = false;
346   pedro.email = "petermese@um.es";
347   pedro.notaMedia = 0.0;

349   pedro.asignaturas = new Asignatura[3];
350   pedro.asignaturas[0] = new Asignatura();
351   pedro.asignaturas[0].nombre = "Programación";
352   pedro.asignaturas[0].creditos = 6;
353   pedro.asignaturas[0].cuatrimestre = 2;
354   pedro.asignaturas[0].calificacion = random(0, 10);

356   pedro.asignaturas[1] = new Asignatura();
357   pedro.asignaturas[1].nombre = "Álgebra";
358   pedro.asignaturas[1].creditos = 6;
359   pedro.asignaturas[1].cuatrimestre = 1;
360   pedro.asignaturas[1].calificacion = random(0, 10);

362   pedro.asignaturas[2] = new Asignatura();
363   pedro.asignaturas[2].nombre = "Matemática discreta";
364   pedro.asignaturas[2].creditos = 6;
365   pedro.asignaturas[2].cuatrimestre = 2;
366   pedro.asignaturas[2].calificacion = random(0, 10);

368   marina.nombre = "Marina";
369   marina.apellidos = "Antelo Gutiérrez";
370   marina.fechaNacimiento = new Fecha();
371   marina.fechaNacimiento.anyo = 2003;
372   marina.fechaNacimiento.mes = 1;
373   marina.fechaNacimiento.dia = 22;
374   marina.edad = calcularEdad(diferenciaFechas(hoy,
          marina.fechaNacimiento));
```

```
375   marina.sexo = true;
376   marina.email = "maguanto@um.es";
377   marina.notaMedia = 0.0;

379   marina.asignaturas = new Asignatura[3];
380   marina.asignaturas[0] = new Asignatura();
381   marina.asignaturas[0].nombre = "Programación";
382   marina.asignaturas[0].creditos = 6;
383   marina.asignaturas[0].cuatrimestre = 2;
384   marina.asignaturas[0].calificacion = random(0, 10);

386   marina.asignaturas[1] = new Asignatura();
387   marina.asignaturas[1].nombre = "Álgebra";
388   marina.asignaturas[1].creditos = 6;
389   marina.asignaturas[1].cuatrimestre = 1;
390   marina.asignaturas[1].calificacion = random(0, 10);

392   marina.asignaturas[2] = new Asignatura();
393   marina.asignaturas[2].nombre = "Matemática discreta";
394   marina.asignaturas[2].creditos = 6;
395   marina.asignaturas[2].cuatrimestre = 2;
396   marina.asignaturas[2].calificacion = random(0, 10);

398   for (int i = 0; i < clase.alumnos.length; i++) {
399     Alumno alumno = clase.alumnos[i];
400     print("El alumno/a "+alumno.nombre+" "
401       +alumno.apellidos+" nació el "
402       +alumno.fechaNacimiento.dia+"/"
403       +alumno.fechaNacimiento.mes+"/"
404       +alumno.fechaNacimiento.anyo
405       +" (por lo tanto tiene "
406       +alumno.edad.anyos+" años, "
407       +alumno.edad.meses+" meses y "
408       +alumno.edad.dias+" días), su correo electrónico es "
            +alumno.email);
409     if (!alumno.sexo) {
410       println(" y es un hombre");
411     } else {
412       println(" y es una mujer");
413     }

415     for (int j = 0; j < alumno.asignaturas.length; j++) {
416       println("En la asignatura "+alumno.asignaturas[j].nombre+" ("
417         +alumno.asignaturas[j].creditos+" créditos, cuatrimestre "
418         +alumno.asignaturas[j].cuatrimestre+") "+alumno.nombre
419         +" tiene un "+alumno.asignaturas[j].calificacion);
420       alumno.notaMedia += alumno.asignaturas[j].calificacion;
421     }
422     alumno.notaMedia = alumno.notaMedia / alumno.asignaturas.length;
423     println("La nota media de todas las asignaturas de "+alumno.nombre
424       +" es: "+alumno.notaMedia);
425     println();
426   }

428   ordenarAlumnosNotaMedia(clase.alumnos, true);
429   println("Si ordenamos los alumnos por su nota media, tenemos:");
```

```
430   for (int i = 0; i < clase.alumnos.length; i++) {
431     Alumno alumno = clase.alumnos[i];
432     println(alumno.nombre+" tiene un "+alumno.notaMedia);
433   }
434   println();

436   Alumno mejorAlumno = mejorAlumnoClase(clase);
437   println("Así las cosas, el mejor alumno de clase es "
438   +mejorAlumno.nombre+" con una nota media de "
          +mejorAlumno.notaMedia+"\n");

440   String[] nombresAsignaturas = {"Álgebra", "Matemática discreta", "
          Programación"};
441   for (int i = 0; i < nombresAsignaturas.length; i++) {
442     mejorAlumno = mejorAlumnoAsignatura(clase,
            nombresAsignaturas[i]);
443     println("El mejor alumno de "+nombresAsignaturas[i]+" es "
            +mejorAlumno.nombre);

445     ordenarAlumnosPorAsignatura(clase.alumnos, true,
            nombresAsignaturas[i]);
446     float[] notas = notasAsignatura(clase, nombresAsignaturas[i]);
447     print("Todas las notas de "+nombresAsignaturas[i]+" son: [");
448     for (int j = 0; j < notas.length-1; j++) {
449       print(notas[j]+", ");
450     }
451     println(notas[notas.length-1]+"]");
452   }
453   println();

455   Edad mediaEdad = mediaEdadClase(clase);
456   println("Por último, la mediad de edad de la clase son "
457   +mediaEdad.anyos+" años, "+mediaEdad.meses+" meses y "
458   +mediaEdad.dias+" días");
459 }
```

7.7 Construye las siguientes funciones:

- La que indique cuántas veces aparece cada dígito (0-9) en un determinado número natural.
- La que indique cuántas veces aparece cada dígito en una lista de números naturales.
- Usa una tercera función para mostrar los resultados.

```
1  /*
2   * Crea una lista de una longitud dada con valores
3   * aleatorios entre un mínimo y un máximo
4   */
5  int[] crearListaAleatoria(int longitud, int min, int max) {
6    if ((longitud < 1) || (max < min)) {
7      return null;
8    }

10   int[] lista = new int[longitud];

12   for (int i = 0; i < longitud; i++) {
13     lista[i] = (int)random(min, max+1);
14   }
```

```
16    return lista;
17  }

19  /*
20   * Calcula la suma de dos vectores dados
21   */
22  int[] sumaVectores(int[] vector1, int[] vector2) {
23    if ((vector1 == null) || (vector2 == null) ||
24          (vector1.length != vector2.length)) {
25      return null;
26    }

28    int[] vectorSuma = new int[vector1.length];
29    for (int i = 0; i < vector1.length; i++) {
30      vectorSuma[i] = vector1[i] + vector2[i];
31    }

33    return vectorSuma;
34  }

36  /*
37   * Cuenta el número de veces que aparece cada
38   * dígito (0-9) en una lista de números dada.
39   */
40  int[] cuentaDigitos(int[] lista) {
41    int[] digitos = new int[10];
42    for (int i = 0; i < digitos.length; i++){
43      digitos[i] = 0;
44    }

46    for (int i = 0; i < lista.length; i++){
47      digitos = sumaVectores(digitos, cuentaDigitos(lista[i]));
48    }

50    return digitos;
51  }

53  /*
54   * Cuenta el número de veces que aparece cada
55   * dígito (0-9) en un número dado.
56   */
57  int[] cuentaDigitos(int numero) {
58    int[] digitos = new int[10];
59    for (int i = 0; i < digitos.length; i++){
60      digitos[i] = 0;
61    }

63    while (numero > 10) {
64      anyadirDigito(digitos, numero%10);
65      numero = numero / 10;
66    }
67    anyadirDigito(digitos, numero);
68    return digitos;
69  }
```

```
/*
 * Incrementa el contador del dígito dado en
 * el array de dígitos dado.
 */
void anyadirDigito(int[] digitos, int digito){
   if ((digito >= 0) && (digito <= 9) &&
       (digitos.length == 10)) {
     digitos[digito]++;
   }
}

/*
 * Muestra por pantalla los valores de una
 * lista dada
 */
void mostrarLista(int[] lista){
   if (lista != null) {
     for (int i = 0; i < lista.length; i++){
       print(lista[i]+" ");
     }
   }
   println();
}

void setup(){
  int longitud = 5;
  int min = 10;
  int max = 100;
  int N = 10;

  for (int i = 0; i < N; i++) {
     int[] miLista = crearListaAleatoria(longitud, min, max);
     mostrarLista(miLista);
     int[] digitos = cuentaDigitos(miLista);
     println("El número de veces que aparece cada dígito en la lista
         es:");
     for (int j = 0; j < digitos.length; j++) {
       println(digitos[j]+" "+j+"'s ");
     }
     println();
  }
}
```

7.8 Dada una cadena de caracteres que representa una secuencia de números naturales separados por espacios (por ejemplo, ‘‘`432 65 854 23 1234`’’), construye las siguientes funciones:

- La que devuelve una lista de cadenas de caracteres, representando cada uno de los números naturales contenidos en la cadena original.
- La que dada una cadena que representa un número natural, la convierte al número en sí mismo. Nota: no se permite el uso de la función `int()` de Processing.
- La que dado un número natural, lo convierte en su representación correspondiente como cadena de caracteres. Nota: no se permiten funciones especiales de Processing, ni “trucos” como concatenar directamente el número dado con la cadena vacía o similar; se espera hacer uso de bucles.
- La que dada la cadena original de secuencia de números naturales, devuelve otra cadena

de caracteres con esos mismos números, pero ordenados en orden ascendente. Por ejemplo, para el caso de la cadena ‘‘432 65 854 23 1234’’, debería devolver ‘‘23 65 432 854 1234’’.

```
/*
 * Crea una cadena de caracteres que representa
 * una lista de enteros de una longitud dada con
 * valores aleatorios entre un mínimo y un máximo.
 */
String crearListaAleatoria(int longitud, int min, int max) {
  if ((longitud < 1) || (max < min)) {
    return null;
  }

  String lista = new String();

  for (int i = 0; i < longitud-1; i++) {
     lista += convertirACadena((int)random(min, max+1)) + " ";
  }
  lista += convertirACadena((int)random(min, max+1));

  return lista;
}

/*
 * Dada una cadena de caracteres que representa una
 * lista de números naturales, devuelve un array de
 * cadenas que representan cada uno de dichos números.
 */
String[] separarNumeros(String listaNumeros) {
   int contadorNumeros = 0;

   for(int i = 0; i < listaNumeros.length(); i++) {
     while((i < listaNumeros.length()) &&
           (listaNumeros.charAt(i) == ' ')){
       i++;
     }
     if (i < listaNumeros.length()) {
       while((i < listaNumeros.length()) &&
             (listaNumeros.charAt(i) != ' ')){
         i++;
       }
       contadorNumeros++;
     }
   }

   String[] numeros = new String[contadorNumeros];
   if (contadorNumeros == 0) {
     return numeros;
   }

   contadorNumeros = 0;
   for(int i = 0; i < listaNumeros.length(); i++) {
     while((i < listaNumeros.length()) &&
           (listaNumeros.charAt(i) == ' ')){
       i++;
```

```
53        }
54        if (i < listaNumeros.length()) {
55          numeros[contadorNumeros] = new String();
56          while((i < listaNumeros.length()) &&
57                (listaNumeros.charAt(i) != ' ')){
58            numeros[contadorNumeros] += listaNumeros.charAt(i);
59            i++;
60          }
61          contadorNumeros++;
62        }
63      }

65      return numeros;
66  }

68  /*
69   * Convierte un carácter en su correspondiente
70   * dígito. Si el carácter no representa un dígito
71   * entonces devuelve -1 como código de error.
72   */
73  int caracterADigito(char caracter){
74    if (caracter == '0') {
75      return 0;
76    } else if (caracter == '1') {
77      return 1;
78    } else if (caracter == '2') {
79      return 2;
80    } else if (caracter == '3') {
81      return 3;
82    } else if (caracter == '4') {
83      return 4;
84    } else if (caracter == '5') {
85      return 5;
86    } else if (caracter == '6') {
87      return 6;
88    } else if (caracter == '7') {
89      return 7;
90    } else if (caracter == '8') {
91      return 8;
92    } else if (caracter == '9') {
93      return 9;
94    } else {
95      return -1;
96    }
97  }

99  /*
100  * Dada una cadena que representa un número natural
101  * devuelve dicho número natural en sí mismo. Si la
102  * cadena no representa un número natural, entonces
103  * devuelve -1 como código de error.
104  */
105 int convertirANumero(String cadenaNumero) {
106   int numero = 0;

108   for (int i = 0; i < cadenaNumero.length(); i++) {
```

```
109       int digito = caracterADigito(cadenaNumero.charAt(i));
110       if (digito == -1) {
111         return -1;
112       }
113       numero += digito * pow(10, cadenaNumero.length()-i-1);
114     }

116     return numero;
117   }

119   /*
120    * Convierte un dígito dado en su correspondiente
121    * carácter. Si el parámetro no representa un
122    * dígito, entonces devuelve el carácter '?' como
123    * código de error.
124    */
125   char digitoACaracter(int digito) {
126     switch(digito) {
127       case 0:
128         return '0';
129       case 1:
130         return '1';
131       case 2:
132         return '2';
133       case 3:
134         return '3';
135       case 4:
136         return '4';
137       case 5:
138         return '5';
139       case 6:
140         return '6';
141       case 7:
142         return '7';
143       case 8:
144         return '8';
145       case 9:
146         return '9';
147       default:
148         return '?';
149     }

151   }

153   /*
154    * Dado un número natural, devuelve la
155    * cadena de caracteres que lo representa.
156    */
157   String convertirACadena(int numero) {
158     String cadenaNumero = new String();

160     while(numero > 10) {
161       cadenaNumero = digitoACaracter(numero%10) + cadenaNumero;
162       numero = numero/10;
163     }
164     cadenaNumero = digitoACaracter(numero) + cadenaNumero;
```

```
166   return cadenaNumero;
167 }

169 /*
170  * Dada una cadena de caracteres que representa una lista
171  * de números naturales, devuelve otra cadena de caracteres
172  * que representa la lista ordenada de dichos naturales de
173  * forma ascendente o descendente (dependiendo del parámetro
174  * ascendente) utilizando el método burbuja.
175  */
176 String ordenarCadenaListaNumeros(String listaNumeros, boolean
        ascendente) {
177   String[] numeros = separarNumeros(listaNumeros);
178   int longitud = numeros.length;

180   do {
181     int nuevaLongitud = 0;
182     for (int i = 1; i <= longitud-1; i++) {
183       if ((ascendente && (convertirANumero(numeros[i-1]) >
              convertirANumero(numeros[i]))) ||
184           (!ascendente && (convertirANumero(numeros[i-1]) <
                convertirANumero(numeros[i])))) {
185         String aux = numeros[i-1];
186         numeros[i-1] = numeros[i];
187         numeros[i] = aux;
188         nuevaLongitud = i;
189       }
190     }
191     longitud = nuevaLongitud;
192   } while(longitud != 0);

194   String listaNumerosOrdenada = new String();
195   for (int i = 0; i < numeros.length-1; i++) {
196     listaNumerosOrdenada += numeros[i] + " ";
197   }
198   listaNumerosOrdenada += numeros[numeros.length-1];

200   return listaNumerosOrdenada;
201 }

203 void setup(){
204   int longitud = 5;
205   int min = 10;
206   int max = 1000;
207   int N = 10;

209   for (int i = 0; i < N; i++) {
210     String listaNumeros = crearListaAleatoria(longitud, min, max);;
211     String[] numeros = separarNumeros(listaNumeros);
212     println("La lista de números \""+listaNumeros+"\" está compuesta
          por los números:");
213     for (int j = 0; j < numeros.length; j++) {
214       String cadenaNumero = numeros[j];
215       print("\""+cadenaNumero+"\" --> ");
216       int numero = convertirANumero(cadenaNumero);
```

```
217       print(numero+" --> ");
218       cadenaNumero = convertirACadena(numero);
219       println("\""+cadenaNumero+"\"");
220     }

222     listaNumeros = ordenarCadenaListaNumeros(listaNumeros, true);
223     println("Y después de ordenarla, la lista queda
            así:\n\""+listaNumeros+"\"\n");
224   }
225 }
```

Tema 8. Recursividad (Soluciones)

8.1 Tema 8: Ejercicios para clase de teoría

8.1 Calcula de forma recursiva la suma de los n-primeros naturales.

```
int sumaPrimerosNaturales(int n) {
  if (n <= 1) {
    return 1;
  }
  return n + sumaPrimerosNaturales(n-1);
}

void setup() {
   int N = 5;
   int min = 10;
   int max = 30;

   for (int i = 0; i < N; i++) {
     int n = (int)random(min, max+1);
     int suma = sumaPrimerosNaturales(n);
     println("La suma de los "+n+" primeros números naturales es "
         +suma);
   }
}
```

8.2 Dada una lista de números enteros, contar el número de veces que aparece un número dado en dicha lista, haciendo uso de la recursividad.

```
/*
 * Crea una lista de una longitud dada con valores
 * aleatorios entre un mínimo y un máximo
 */
int[] crearListaAleatoria(int longitud, int min, int max) {
  if ((longitud < 1) || (max < min)) {
    return null;
  }
```

```
10    int[] lista = new int[longitud];

12    for (int i = 0; i < longitud; i++) {
13       lista[i] = (int)random(min, max+1);
14    }

16    return lista;
17  }

19  /*
20   * Muestra por pantalla los valores de una
21   * lista dada
22   */
23  void mostrarLista(int[] lista){
24     if (lista != null) {
25       for (int i = 0; i < lista.length; i++){
26         print(lista[i]+" ");
27       }
28     }
29     println();
30  }

32  int contarOcurrencias(int[] lista, int numero) {
33    return contarOcurrenciasRecursiva(lista, numero, lista.length-1);
34  }

36  int contarOcurrenciasRecursiva(int[] lista, int numero, int indice) {
37    if (indice == 0) {
38       if (lista[indice] == numero) {
39         return 1;
40       } else {
41         return 0;
42       }
43    } else {
44       if (lista[indice] == numero) {
45         return 1+contarOcurrenciasRecursiva(lista, numero, indice-1);
46       } else {
47         return contarOcurrenciasRecursiva(lista, numero, indice-1);
48       }
49    }
50  }

52  void setup() {
53     int N = 10;
54     int longitud = 8;
55     int min = 2;
56     int max = 3;

58     for (int i = 0; i < N; i++) {
59       int[] lista = crearListaAleatoria(longitud, min, max);
60       int n = (int)random(min, max+1);
61       int contador = contarOcurrencias(lista, n);
62       print("El número "+n+" aparece "+contador+" veces en la lista: "
             );
63       mostrarLista(lista);
```

```
64        println();
65      }
66  }
```

8.3 Todo número n es primo si no hay divisor entre el 2 y $n/2$. Dado un número n indica, usando recursividad, si es primo o no.

```
1  boolean esPrimo(int n) {
2    return esPrimoRecursividad(n, 2);
3  }

5  boolean esPrimoRecursividad(int n, int divisor) {
6    if (divisor > n/2) {
7      return true;
8    }
9    if (n%divisor == 0) {
10     return false;
11   }

13   return esPrimoRecursividad(n, divisor+1);
14 }

16 void setup() {
17    int N = 10;
18    int min = 2;
19    int max = 100;

21    for (int i = 0; i < N; i++) {
22      int n = (int)random(min, max+1);
23      if (esPrimo(n)) {
24        println("El número "+n+" es primo");
25      } else {
26        println("El número "+n+" no es primo");
27      }
28    }
29 }
```

8.2 Tema 8: Ejercicios para clase de prácticas

8.4 Calcula de forma recursiva el factorial de un número n.

```
int factorial(int n) {
  if (n <= 1) {
    return 1;
  }
  return n * factorial(n-1);
}

void setup() {
   int N = 5;
   int min = 3;
   int max = 15;

   for (int i = 0; i < N; i++) {
     int n = (int)random(min, max+1);
     int factorial = factorial(n);
     println("El factorial de "+n+" es "+factorial);
   }
}
```

8.5 La sucesión de Fibonacci es:

$$1, 1, 2, 3, 5, 8, 13 \ldots$$

Usando recursividad, indica cuál es el i-ésimo término de la sucesión de Fibonacci.

```
int fibonacci(int n) {
  if (n <= 2) {
    return 1;
  }
  return fibonacci(n-1) + fibonacci(n-2);
}

void setup() {
   int N = 5;
   int min = 1;
   int max = 15;

   for (int i = 0; i < N; i++) {
     int n = (int)random(min, max+1);
     int fibonacci = fibonacci(n);
     println("El término número "+n+" de la sucesión de Fibonacci es "
         +fibonacci);
   }

   println();
   println("La sucesión de Fibonacci para los primeros "+max+"
       términos es:");
   for (int i = 1; i <= max; i++) {
     print(fibonacci(i)+" ");
   }
}
```

8.3 Tema 8: Ejercicios para casa

8.6 Haz una función recursiva que muestre por pantalla los dígitos de un número *n* dado.

```
void mostrarDigitos(int n) {
  if (n < 10) {
    print(n+" ");
    return;
  }
  mostrarDigitos(n/10);
  print((n%10)+" ");
}

void setup() {
   int N = 10;
   int min = 100;
   int max = 10000;

   for (int i = 0; i < N; i++) {
     int n = (int)random(min, max+1);
     print("Los dígitos de "+n+" son: ");
     mostrarDigitos(n);
     println();
   }
}
```

8.7 Haz una función recursiva que devuelva la suma de los dígitos que componen un número *n* dado.

```
int sumarDigitos(int n) {
  if (n < 10) {
    return n;
  }
  return (n%10) + sumarDigitos(n/10);
}

void setup() {
   int N = 10;
   int min = 100;
   int max = 10000;

   for (int i = 0; i < N; i++) {
     int n = (int)random(min, max+1);
     int suma = sumarDigitos(n);
     println("La suma de los dígitos de "+n+" es: "+suma);
   }
}
```

8.8 Dada una lista de números, hacer una función recursiva que devuelva la misma lista, pero en orden inverso.

```
/*
 * Crea una lista de una longitud dada con valores
 * aleatorios entre un mínimo y un máximo
 */
int[] crearListaAleatoria(int longitud, int min, int max) {
  if ((longitud < 1) || (max < min)) {
```

```
7       return null;
8     }

10    int[] lista = new int[longitud];

12    for (int i = 0; i < longitud; i++) {
13       lista[i] = (int)random(min, max+1);
14    }

16    return lista;
17  }

19  /*
20   * Muestra por pantalla los valores de una
21   * lista dada
22   */
23  void mostrarLista(int[] lista){
24     if (lista != null) {
25       for (int i = 0; i < lista.length; i++){
26         print(lista[i]+" ");
27       }
28     }
29     println();
30  }

32  /*
33   * Dado un número y una lista, devuelve una nueva
34   * lista con los mismos elementos que la lista dada,
35   * pero conteniendo el número dado al principio de
36   * la nueva lista
37   */
38  int[] anyadirElemento(int numero, int[] lista){
39    int[] nuevaLista = new int[lista.length+1];

41    nuevaLista[0] = numero;
42    for (int i = 0; i < lista.length;  i++) {
43      nuevaLista[i+1] = lista[i];
44    }
45    return nuevaLista;
46  }

48  int[] listaInversa(int[] lista) {
49    return listaInversaRecursiva(lista, lista.length-1);
50  }

52  int[] listaInversaRecursiva(int[] lista, int indice) {
53     if (indice == 0) {
54        int[] nuevaLista = new int[1];
55        nuevaLista[indice] = lista[indice];
56        return nuevaLista;
57     }
58     return anyadirElemento(lista[indice], listaInversaRecursiva(lista,
           indice-1));
59  }

61  void setup() {
```

```
  int N = 10;
  int longitud = 6;
  int min = 2;
  int max = 100;

  for (int i = 0; i < N; i++) {
    int[] lista = crearListaAleatoria(longitud, min, max);
    print("La lista inversa de ");
    mostrarLista(lista);
    print("es ");
    mostrarLista(listaInversa(lista));
    println();
  }
}
```

8.9 Dada una lista de números, hacer una función recursiva que devuelva la suma de sus elementos.

```
void setup() {
  // Ejemplo de uso
  int[] numeros = {1, 2, 3, 4, 5};
  int suma = sumaLista(numeros);
  println("La suma de los elementos es: " + suma);
}

int sumaLista(int[] lista) {
  return sumaRecursiva(lista, 0, lista.length - 1);
}

int sumaRecursiva(int[] lista, int inicio, int fin) {
  if (inicio == fin) {
    return lista[inicio];
  } else {
    int medio = (inicio + fin) / 2;
    int sumaIzquierda = sumaRecursiva(lista, inicio, medio);
    int sumaDerecha = sumaRecursiva(lista, medio + 1, fin);
    return sumaIzquierda + sumaDerecha;
  }
}
```

8.10 Haz una función recursiva para calcular la potencia de un número a^b.

```
void setup() {
  // Ejemplo de uso
  int base = 2;
  int exponente = 5;
  int resultado = potencia(base, exponente);
  println(base + " elevado a la potencia " + exponente + " es igual a
      " + resultado);
}

int potencia(int base, int exponente) {
  if (exponente == 0) {
    return 1;
  } else {
    return base * potencia(base, exponente - 1);
  }
}
```

Tema 10. Programación orientada a objetos (Solucione

10.1 Tema 10: Ejercicios para clase de teoría

10.1 Crea una clase para representar círculos definidos por su radio y su color. Permitir calcular el área y el perímetro de un círculo.

```
1  class Circulo {
2    String colorCirculo;
3    float radio;
4
5    float perimetro() {
6      return 2*PI*radio;
7    }
8
9    float area() {
10     return PI*radio*radio;
11   }
12 }
13
14 void setup() {
15   int N = 10;
16   float minRadio = 1.0;
17   float maxRadio = 10.0;
18   String[] colores = {"Rojo", "Amarillo", "Verde", "Azul", "Naranja"};
19
20   for (int i = 0; i < N; i++) {
21     Circulo circulo = new Circulo();
22     circulo.radio = random(minRadio, maxRadio);
23     int indiceColor = (int)random(0, colores.length);
24     circulo.colorCirculo = colores[indiceColor];
25
26     float area = circulo.area();
27     float perimetro = circulo.perimetro();
28
29     println("El círculo de radio "+circulo.radio+" y color "+
30     circulo.colorCirculo+" tiene un perímetro de "+perimetro+
```

```
    " y un área de "+area+"\n");
  }
}
```

10.2 A partir del ejercicio **10.1**, crea una clase para representar cilindros, que permitan calcular su superficie y su volumen.

```
class Circulo {
  String colorCirculo;
  float radio;

  float perimetro() {
    return 2*PI*radio;
  }

  float area() {
    return PI*radio*radio;
  }
}

class Cilindro {
  Circulo base;
  float altura;

  float superficie() {
    return base.perimetro()*altura + 2*base.area();
  }

  float volumen() {
    return base.area()*altura;
  }
}

void setup() {
  int N = 10;

  float minRadio = 1.0;
  float maxRadio = 10.0;
  String[] colores = {"Rojo", "Amarillo", "Verde", "Azul", "Naranja"};

  float minAltura = 2.0;
  float maxAltura = 15.0;

  for (int i = 0; i < N; i++) {
    Circulo circulo = new Circulo();
    circulo.radio = random(minRadio, maxRadio);
    int indiceColor = (int)random(0, colores.length);
    circulo.colorCirculo = colores[indiceColor];

    Cilindro cilindro = new Cilindro();
    cilindro.base = circulo;
    cilindro.altura = random(minAltura, maxAltura);

    float superficie = cilindro.superficie();
    float volumen = cilindro.volumen();

    println("El cilindro con base de radio "+cilindro.base.radio+
```

```
    " y altura "+cilindro.altura+" tiene una superficie de "+
    superficie+" y un volumen de "+volumen+"\n");
  }
}
```

10.3 Crea una clase para representar un punto (x, y) en el eje de coordenadas cartesianas. Permitir calcular la distancia euclídea entre dos puntos.

```
class Punto {
  float coordX;
  float coordY;

  float distanciaEuclidea(Punto punto) {
    return sqrt(pow(coordX - punto.coordX, 2) + pow(coordY -
        punto.coordY, 2));
  }
}

void setup() {
  int N = 10;
  float min = 1.0;
  float max = 10.0;

  for (int i = 0; i < N; i++) {
    Punto punto1 = new Punto();
    punto1.coordX = random(min, max);
    punto1.coordY = random(min, max);

    Punto punto2 = new Punto();
    punto2.coordX = random(min, max);
    punto2.coordY = random(min, max);

    float distancia = punto1.distanciaEuclidea(punto2);

    println("La distancia euclídea del punto ("+punto1.coordX+", "+
    punto1.coordY+") al punto ("+punto2.coordX+", "+punto2.coordY+
    ") es "+distancia+"\n");
  }
}
```

10.4 Amplía el ejercicio **10.1** para permitir saber si dos círculos se solapan o no.

```
class Punto {
  float coordX;
  float coordY;

  float distanciaEuclidea(Punto punto) {
    return sqrt(pow(coordX - punto.coordX, 2) + pow(coordY -
        punto.coordY, 2));
  }
}

class Circulo {
  String colorCirculo;
  float radio;
  Punto centro;
```

```
15    float perimetro() {
16      return 2*PI*radio;
17    }

19    float area() {
20      return PI*radio*radio;
21    }

23    boolean solapa(Circulo circulo) {
24      return ((radio + circulo.radio) >=
25              centro.distanciaEuclidea(circulo.centro));
26    }
27  }

29  void setup() {
30    int N = 10;
31    float min = 1.0;
32    float max = 15.0;

34    float minRadio = 1.0;
35    float maxRadio = 10.0;
36    String[] colores = {"Rojo", "Amarillo", "Verde", "Azul", "Naranja"};

38    for (int i = 0; i < N; i++) {
39      Circulo circulo1 = new Circulo();
40      circulo1.radio = random(minRadio, maxRadio);
41      int indiceColor = (int)random(0, colores.length);
42      circulo1.colorCirculo = colores[indiceColor];

44      Punto punto1 = new Punto();
45      punto1.coordX = random(min, max);
46      punto1.coordY = random(min, max);
47      circulo1.centro = punto1;

49      Circulo circulo2 = new Circulo();
50      circulo2.radio = random(minRadio, maxRadio);
51      indiceColor = (int)random(0, colores.length);
52      circulo2.colorCirculo = colores[indiceColor];

54      Punto punto2 = new Punto();
55      punto2.coordX = random(min, max);
56      punto2.coordY = random(min, max);
57      circulo2.centro = punto2;

59      print("El círculo de color "+circulo1.colorCirculo+", con radio "+
60      circulo1.radio+" y centro en ("+circulo1.centro.coordX+", "+
61      circulo1.centro.coordY+") y el círculo de color "
            +circulo2.colorCirculo+
62      ", con radio "+circulo2.radio+" y centro en
            ("+circulo2.centro.coordX+
63      ", "+circulo2.centro.coordY+") ");

65      if (circulo1.solapa(circulo2)) {
66        println("sí se solapan\n");
67      } else {
68        println("no se solapan\n");
```

```
69        }
70      }
71  }
```

10.2 Tema 10: Ejercicios para clase de prácticas

10.5 Una Pila es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se apilan y desapilan elementos. Los elementos se apilan uno detrás de otro y, a la hora de desapilar, se elimina el último elemento que haya sido apilado.

Construye una clase que represente una Pila de números naturales de un tamaño máximo dado, con las operaciones para apilar, desapilar, indicar si la pila está vacía y mostrar los elementos de la pila por pantalla.

```
1  class Pila {
2    int[] contenido;
3    int numElementos = 0;

5    void apilar(int numero) {
6      if ((contenido != null) && (numElementos < contenido.length)) {
7        contenido[numElementos] = numero;
8        numElementos++;
9      }
10   }

12   int desapilar(){
13     if (numElementos > 0) {
14       numElementos--;
15       return contenido[numElementos];
16     } else {
17       return -1;
18     }
19   }

21   boolean estaVacia() {
22     return (numElementos == 0);
23   }

25   void imprimir(){
26     print("[");
27     for(int i = 0; i < numElementos-1; i++) {
28       print(contenido[i]+", ");
29     }
30     if ((contenido != null) && (numElementos > 0)){
31       print(contenido[numElementos-1]);
32     }
33     print("]");
34   }
35 }


38 void setup() {
39   int N = 10;
40   float min = 1;
41   float max = 100;

43   float minCapacidad = 5;
44   float maxCapacidad = 10;

46   for (int i = 0; i < N; i++) {
```

```
47      Pila pila = new Pila();
48      int capacidad = (int)random(minCapacidad, maxCapacidad);
49      pila.contenido = new int[capacidad];

51      for (int j = 0; j < capacidad; j++) {
52        int numero = (int)random(min, max);
53        pila.apilar(numero);
54        pila.imprimir();
55        println();
56      }
57      println();

59      while(!pila.estaVacia()) {
60        int numero = pila.desapilar();
61        pila.imprimir();
62        println(" --> "+numero);
63      }
64      println("\n");
65    }
66  }
```

10.6 A partir del ejercicio **10.3**, construye una clase para representar una recta en el plano definida a partir de dos puntos A y B, con $A \neq B$. Permitir las operaciones para saber si dos rectas son paralelas o no, para obtener el punto de intersección (en caso de que no sean paralelas), para conocer la distancia entre dos rectas (en caso de que sean paralelas) y para conocer la distancia de un punto a una recta.

```
1   class Punto {
2     float coordX;
3     float coordY;

5     float distanciaEuclidea(Punto punto) {
6       return sqrt(pow(coordX - punto.coordX, 2) + pow(coordY -
            punto.coordY, 2));
7     }

9     String toString(){
10      return "("+coordX+", "+coordY+")";
11    }
12  }

14  class Recta {
15    Punto a;
16    Punto b;

18    boolean paralela(Recta recta) {
19      float pendiente1 = (b.coordY - a.coordY)/(b.coordX - a.coordX);
20      float pendiente2 = (recta.b.coordY -
            recta.a.coordY)/(recta.b.coordX - recta.a.coordX);
21      return (pendiente1 == pendiente2);
22    }

24    Punto puntoInterseccion(Recta recta) {
25      float pendiente1 = (b.coordY - a.coordY)/(b.coordX - a.coordX);
26      float terminoInd1 = a.coordY - pendiente1*a.coordX;
27      float pendiente2 = (recta.b.coordY -
```

```
        recta.a.coordY)/(recta.b.coordX - recta.a.coordX);
28    float terminoInd2 = recta.a.coordY - pendiente2*recta.a.coordX;

30    if (pendiente1 == pendiente2) {
31      return null; //Las dos rectas son paralelas
32    }
33    Punto interseccion = new Punto();
34    interseccion.coordX =
          (terminoInd2-terminoInd1)/(pendiente1-pendiente2);
35    interseccion.coordY = pendiente1*interseccion.coordX +
          terminoInd1;
36    return interseccion;
37  }

39  float distanciaPunto(Punto punto) {
40    float pendiente = (b.coordY - a.coordY)/(b.coordX - a.coordX);
41    float terminoInd = a.coordY - pendiente*a.coordX;

43    return abs(pendiente*punto.coordX - punto.coordY + terminoInd)/
44             sqrt(pow(pendiente, 2)+1);
45  }

47  float distanciaRecta(Recta recta) {
48    return recta.distanciaPunto(a);
49  }
50 }

52 void setup() {
53   int N = 10;
54   float min = 1.0;
55   float max = 10.0;

57   for (int i = 0; i < N; i++) {
58     Punto punto1 = new Punto();
59     punto1.coordX = random(min, max);
60     punto1.coordY = random(min, max);

62     Punto punto2 = new Punto();
63     punto2.coordX = random(min, max);
64     punto2.coordY = random(min, max);

66     Recta recta1 = new Recta();
67     recta1.a = punto1;
68     recta1.b = punto2;

70     Punto punto3 = new Punto();
71     punto3.coordX = random(min, max);
72     punto3.coordY = random(min, max);

74     Punto punto4 = new Punto();
75     punto4.coordX = random(min, max);
76     punto4.coordY = random(min, max);

78     Recta recta2 = new Recta();
79     recta2.a = punto3;
80     recta2.b = punto4;
```

```
82      println("La recta 1 pasa por los puntos "+punto1.toString()+
83      " y "+punto2.toString());
84      println("La recta 2 pasa por los puntos "+punto3.toString()+
85      " y "+punto4.toString());
86      if (recta1.paralela(recta2)) {
87        float distancia = recta1.distanciaRecta(recta2);
88        println("Las rectas son paralelas y la distancia entre ellas es
              "+distancia);
89      } else {
90        Punto punto5 = recta1.puntoInterseccion(recta2);
91        println("Las rectas son secantes e intersectan en el punto "+
92                  punto5.toString());
93      }
94      println();
95    }
96  }
```

10.3 Tema 10: Ejercicios para casa

10.7 Una Cola es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se añaden y se eliminan elementos. Los elementos se añaden uno detrás de otro y, a la hora de eliminar un elemento, se elimina más antiguo, y el resto de elementos avanzan una posición en la cola.
Construye una clase que represente una Cola de números naturales de un tamaño máximo dado, con las operaciones de añadir elemento, eliminar elemento, indicar si la cola está vacía y mostrar los elementos de la cola por pantalla.

```
1  class Cola {
2    int[] contenido;
3    int numElementos = 0;
4
5    void anyadir(int numero) {
6      if ((contenido != null) && (numElementos < contenido.length)) {
7        contenido[numElementos] = numero;
8        numElementos++;
9      }
10   }
11
12   int eliminar(){
13     if (numElementos > 0) {
14       int numero = contenido[0];
15       for (int i = 0; i < numElementos-1; i++) {
16         contenido[i] = contenido[i+1];
17       }
18       numElementos--;
19       return numero;
20     } else {
21       return -1;
22     }
23   }
24
25   boolean estaVacia() {
26     return (numElementos == 0);
27   }
28
29   void imprimir(){
30     print("[");
31     for(int i = 0; i < numElementos-1; i++) {
32       print(contenido[i]+", ");
33     }
34     if ((contenido != null) && (numElementos > 0)){
35       print(contenido[numElementos-1]);
36     }
37     print("]");
38   }
39 }
40
41
42 void setup() {
43   int N = 10;
44   float min = 1;
45   float max = 100;
```

```
47    float minCapacidad = 5;
48    float maxCapacidad = 10;

50    for (int i = 0; i < N; i++) {
51      Cola cola = new Cola();
52      int capacidad = (int)random(minCapacidad, maxCapacidad);
53      cola.contenido = new int[capacidad];

55      for (int j = 0; j < capacidad; j++) {
56        int numero = (int)random(min, max);
57        cola.anyadir(numero);
58        cola.imprimir();
59        println();
60      }
61      println();

63      while(!cola.estaVacia()) {
64        int numero = cola.eliminar();
65        cola.imprimir();
66        println(" --> "+numero);
67      }
68      println("\n");
69    }
70  }
```

10.8 Modifica el código del ejercicio **10.7** para que la cola no tenga un tamaño máximo.

```
1   class Cola {
2     int[] contenido;

4     void anyadir(int numero) {
5       if (contenido == null) {
6         contenido = new int[1];
7       } else {
8         int[] aux = new int[contenido.length+1];
9         for (int i = 0; i < contenido.length; i++) {
10          aux[i] = contenido[i];
11        }
12        contenido = aux;
13      }
14      contenido[contenido.length-1] = numero;
15    }

17    int eliminar(){
18      if ((contenido != null) && (contenido.length > 0)) {
19        int numero = contenido[0];
20        int[] aux = new int[contenido.length-1];
21        for (int i = 0; i < contenido.length-1; i++) {
22          aux[i] = contenido[i+1];
23        }
24        contenido = aux;
25        return numero;
26      } else {
27        return -1;
28      }
29    }
```

```
  boolean estaVacia() {
    return ((contenido == null) || (contenido.length == 0));
  }

  void imprimir(){
    print("[");
    for(int i = 0; i < contenido.length-1; i++) {
      print(contenido[i]+", ");
    }
    if ((contenido != null) && (contenido.length > 0)){
      print(contenido[contenido.length-1]);
    }
    print("]");
  }
}


void setup() {
  int N = 10;
  float min = 1;
  float max = 100;

  float minCapacidad = 5;
  float maxCapacidad = 10;

  for (int i = 0; i < N; i++) {
    Cola cola = new Cola();
    int capacidad = (int)random(minCapacidad, maxCapacidad);

    for (int j = 0; j < capacidad; j++) {
      int numero = (int)random(min, max);
      cola.anyadir(numero);
      cola.imprimir();
      println();
    }
    println();

    while(!cola.estaVacia()) {
      int numero = cola.eliminar();
      cola.imprimir();
      println(" --> "+numero);
    }
    println("\n");
  }
}
```

10.9 Modifica el código del ejercicio **7.6** para utilizar clases, objetos y métodos en lugar de registros.

```
class Clase {
  int curso;
  Alumno[] alumnos;

  Alumno mejorAlumnoClase() {
    float mejorNota = -1;
    Alumno mejorAlumno = null;

    for (int i = 0; i < alumnos.length; i++) {
```

```
10        Alumno alumno = alumnos[i];
11        if (alumno.notaMedia > mejorNota) {
12          mejorNota = alumno.notaMedia;
13          mejorAlumno = alumno;
14        }
15      }
16      return mejorAlumno;
17    }

19    Alumno mejorAlumnoAsignatura(String nombreAsignatura) {
20      float mejorNota = -1;
21      Alumno mejorAlumno = null;

23      for (int i = 0; i < alumnos.length; i++) {
24        Alumno alumno = alumnos[i];
25        for (int j = 0; j < alumno.asignaturas.length; j++) {
26          Asignatura asignatura = alumno.asignaturas[j];
27          if ((asignatura.nombre.equals(nombreAsignatura))
28           &&( asignatura.calificacion > mejorNota)) {
29            mejorNota = asignatura.calificacion;
30            mejorAlumno = alumno;
31          }
32        }
33      }
34      return mejorAlumno;
35    }

37    float[] notasAsignatura(String nombreAsignatura) {
38      int numNotas = 0;

40      for (int i = 0; i < alumnos.length; i++) {
41        Alumno alumno = alumnos[i];
42        for (int j = 0; j < alumno.asignaturas.length; j++) {
43          Asignatura asignatura = alumno.asignaturas[j];
44          if (asignatura.nombre.equals(nombreAsignatura)) {
45            numNotas++;
46          }
47        }
48      }

50      float[] notas = new float[numNotas];
51      numNotas = 0;

53      for (int i = 0; i < alumnos.length; i++) {
54        Alumno alumno = alumnos[i];
55        for (int j = 0; j < alumno.asignaturas.length; j++) {
56          Asignatura asignatura = alumno.asignaturas[j];
57          if (asignatura.nombre.equals(nombreAsignatura)) {
58            notas[numNotas] = asignatura.calificacion;
59            numNotas++;
60          }
61        }
62      }

64      return notas;
65    }
```

```
67    Edad mediaEdadClase() {
68      int mediaEdadEnDias = 0;
69      for (int i = 0; i < alumnos.length; i++) {
70        Alumno alumno = alumnos[i];
71        mediaEdadEnDias += alumno.edad.edadADias();
72      }
73      mediaEdadEnDias = mediaEdadEnDias / alumnos.length;

75      return calcularEdad(mediaEdadEnDias);
76    }

78    void ordenarAlumnosNotaMedia(boolean ascendente) {
79       int longitud = alumnos.length;

81       do {
82         int nuevaLongitud = 0;
83         for (int i = 1; i <= longitud-1; i++) {
84           if ((ascendente && (alumnos[i-1].notaMedia >
                 alumnos[i].notaMedia)) ||
85               (!ascendente && (alumnos[i-1].notaMedia <
                     alumnos[i].notaMedia))) {
86              Alumno aux = alumnos[i-1];
87              alumnos[i-1] = alumnos[i];
88              alumnos[i] = aux;
89              nuevaLongitud = i;
90           }
91         }
92         longitud = nuevaLongitud;
93       } while(longitud != 0);
94    }

96    void ordenarAlumnosPorAsignatura(boolean ascendente, String
          nombreAsignatura) {
97       int longitud = alumnos.length;
98       float[] calificaciones = notasAsignatura(nombreAsignatura);
99       do {
100        int nuevaLongitud = 0;
101        for (int i = 1; i <= longitud-1; i++) {
102          if ((ascendente && (calificaciones[i-1] >
                 calificaciones[i])) ||
103              (!ascendente && (calificaciones[i-1] <
                     calificaciones[i]))) {
104             Alumno aux = alumnos[i-1];
105             alumnos[i-1] = alumnos[i];
106             alumnos[i] = aux;
107             float aux2 = calificaciones[i-1];
108             calificaciones[i-1] = calificaciones[i];
109             calificaciones[i] = aux2;
110             nuevaLongitud = i;
111          }
112        }
113        longitud = nuevaLongitud;
114      } while(longitud != 0);
115   }
116 }
```

```
118 class Fecha {
119   int dia;
120   int mes;
121   int anyo;

123   int diferenciaFechas(Fecha fecha) {
124     int numDiasFecha1 = anyo*365 + (mes-1)*30 + dia;
125     int numDiasFecha2 = fecha.anyo*365 + (fecha.mes-1)*30 + fecha.dia;

127     if (numDiasFecha2 > numDiasFecha1) {
128       return -1;
129     }

131     return numDiasFecha1 - numDiasFecha2;
132   }
133 }

135 class Edad {
136   int anyos;
137   int meses;
138   int dias;

140   /*
141    * Convierte una edad en su equivalente
142    * en días.
143    */
144   int edadADias() {
145      return anyos*365 + (meses-1)*30 + dias;
146   }
147 }

149 class Asignatura {
150   String nombre;
151   int creditos;
152   int cuatrimestre;
153   float calificacion;
154 }

156 class Alumno {
157   String nombre;
158   String apellidos;
159   Fecha fechaNacimiento;
160   Edad edad;
161   boolean sexo; //True para mujer y false para hombre
162   String email;

164   Asignatura[] asignaturas;
165   float notaMedia;
166 }


169 /*
170  * Calcula una edad (años, meses y días)
171  * a partir de un número total de días
172  */
```

```
173 Edad calcularEdad(int numeroDias) {
174   Edad edad = new Edad();

176   edad.anyos = numeroDias / 365;
177   edad.meses = (numeroDias % 365) / 30;
178   edad.dias = (numeroDias % 365) % 30;

180   return edad;
181 }


184 void setup() {
185   Clase clase = new Clase();
186   clase.curso = 1;
187   clase.alumnos = new Alumno[5];

189   Fecha hoy = new Fecha();
190   hoy.anyo = 2022;
191   hoy.mes = 3;
192   hoy.dia = 23;

194   Alumno alberto = new Alumno();
195   Alumno manuel = new Alumno();
196   Alumno sofia = new Alumno();
197   Alumno pedro = new Alumno();
198   Alumno marina = new Alumno();

200   clase.alumnos[0] = alberto;
201   clase.alumnos[1] = manuel;
202   clase.alumnos[2] = sofia;
203   clase.alumnos[3] = pedro;
204   clase.alumnos[4] = marina;

206   alberto.nombre = "Alberto";
207   alberto.apellidos = "García Martínez";
208   alberto.fechaNacimiento = new Fecha();
209   alberto.fechaNacimiento.anyo = 2001;
210   alberto.fechaNacimiento.mes = 7;
211   alberto.fechaNacimiento.dia = 20;
212   alberto.edad =
          calcularEdad(hoy.diferenciaFechas(alberto.fechaNacimiento));
213   alberto.sexo = false;
214   alberto.email = "alberto.garcia-martinez@um.es";
215   alberto.notaMedia = 0.0;

217   alberto.asignaturas = new Asignatura[3];
218   alberto.asignaturas[0] = new Asignatura();
219   alberto.asignaturas[0].nombre = "Programación";
220   alberto.asignaturas[0].creditos = 6;
221   alberto.asignaturas[0].cuatrimestre = 2;
222   alberto.asignaturas[0].calificacion = random(0, 10);

224   alberto.asignaturas[1] = new Asignatura();
225   alberto.asignaturas[1].nombre = "Álgebra";
226   alberto.asignaturas[1].creditos = 6;
227   alberto.asignaturas[1].cuatrimestre = 1;
```

```
228   alberto.asignaturas[1].calificacion = random(0, 10);

230   alberto.asignaturas[2] = new Asignatura();
231   alberto.asignaturas[2].nombre = "Matemática discreta";
232   alberto.asignaturas[2].creditos = 6;
233   alberto.asignaturas[2].cuatrimestre = 2;
234   alberto.asignaturas[2].calificacion = random(0, 10);

236   manuel.nombre = "Manuel";
237   manuel.apellidos = "Buitrago Gil";
238   manuel.fechaNacimiento = new Fecha();
239   manuel.fechaNacimiento.anyo = 2001;
240   manuel.fechaNacimiento.mes = 12;
241   manuel.fechaNacimiento.dia = 2;
242   manuel.edad =
          calcularEdad(hoy.diferenciaFechas(manuel.fechaNacimiento));
243   manuel.sexo = false;
244   manuel.email = "manuel.bg@um.es";
245   manuel.notaMedia = 0.0;

247   manuel.asignaturas = new Asignatura[3];
248   manuel.asignaturas[0] = new Asignatura();
249   manuel.asignaturas[0].nombre = "Programación";
250   manuel.asignaturas[0].creditos = 6;
251   manuel.asignaturas[0].cuatrimestre = 2;
252   manuel.asignaturas[0].calificacion = random(0, 10);

254   manuel.asignaturas[1] = new Asignatura();
255   manuel.asignaturas[1].nombre = "Álgebra";
256   manuel.asignaturas[1].creditos = 6;
257   manuel.asignaturas[1].cuatrimestre = 1;
258   manuel.asignaturas[1].calificacion = random(0, 10);

260   manuel.asignaturas[2] = new Asignatura();
261   manuel.asignaturas[2].nombre = "Matemática discreta";
262   manuel.asignaturas[2].creditos = 6;
263   manuel.asignaturas[2].cuatrimestre = 2;
264   manuel.asignaturas[2].calificacion = random(0, 10);

266   sofia.nombre = "Sofía";
267   sofia.apellidos = "López Pérez";
268   sofia.fechaNacimiento = new Fecha();
269   sofia.fechaNacimiento.anyo = 2002;
270   sofia.fechaNacimiento.mes = 5;
271   sofia.fechaNacimiento.dia = 18;
272   sofia.edad =
          calcularEdad(hoy.diferenciaFechas(sofia.fechaNacimiento));
273   sofia.sexo = true;
274   sofia.email = "sofialp@um.es";
275   sofia.notaMedia = 0.0;

277   sofia.asignaturas = new Asignatura[3];
278   sofia.asignaturas[0] = new Asignatura();
279   sofia.asignaturas[0].nombre = "Programación";
280   sofia.asignaturas[0].creditos = 6;
281   sofia.asignaturas[0].cuatrimestre = 2;
```

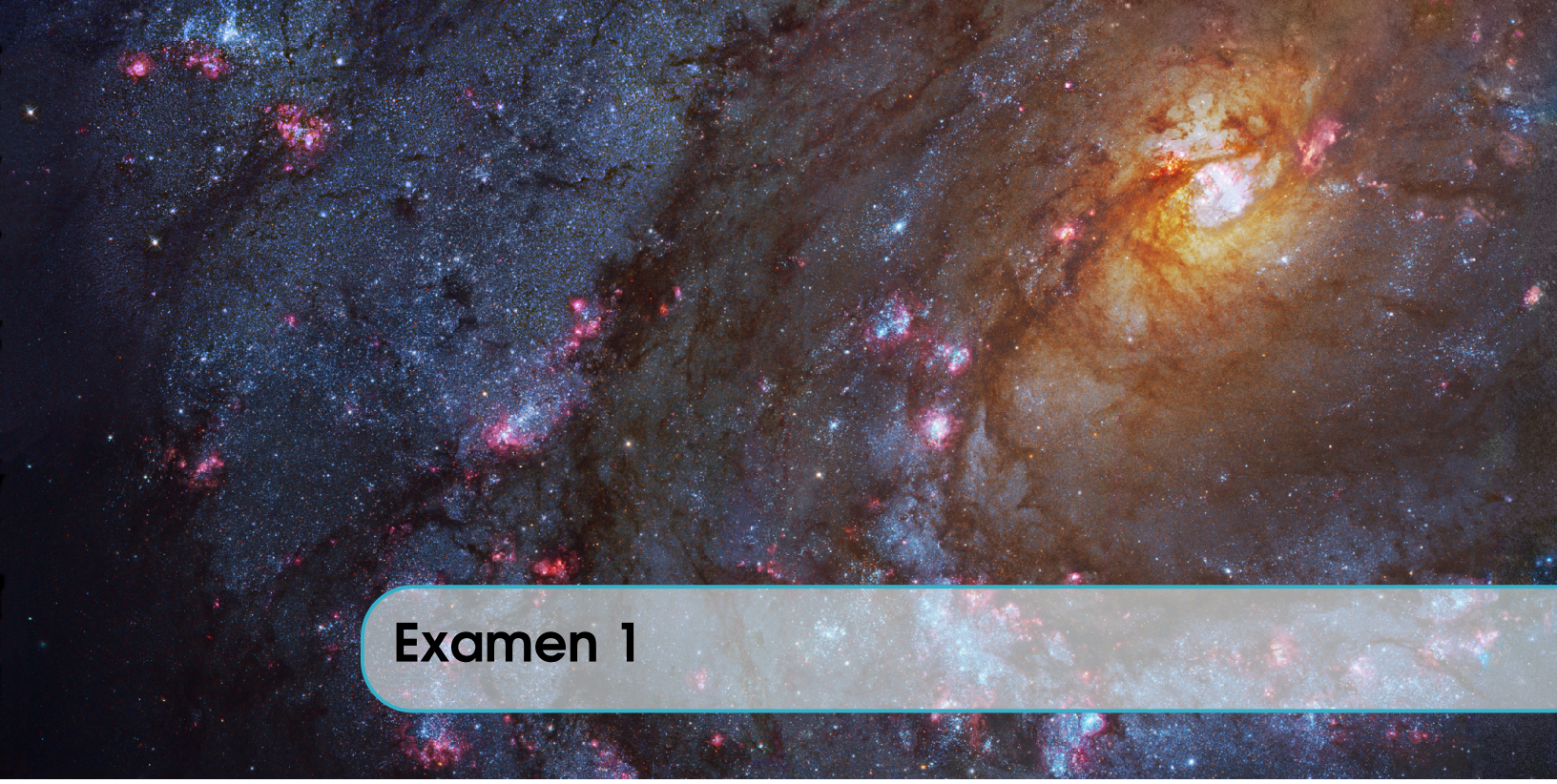
```
282   sofia.asignaturas[0].calificacion = random(0, 10);

284   sofia.asignaturas[1] = new Asignatura();
285   sofia.asignaturas[1].nombre = "Álgebra";
286   sofia.asignaturas[1].creditos = 6;
287   sofia.asignaturas[1].cuatrimestre = 1;
288   sofia.asignaturas[1].calificacion = random(0, 10);

290   sofia.asignaturas[2] = new Asignatura();
291   sofia.asignaturas[2].nombre = "Matemática discreta";
292   sofia.asignaturas[2].creditos = 6;
293   sofia.asignaturas[2].cuatrimestre = 2;
294   sofia.asignaturas[2].calificacion = random(0, 10);

296   pedro.nombre = "Pedro";
297   pedro.apellidos = "Meseguer Morales";
298   pedro.fechaNacimiento = new Fecha();
299   pedro.fechaNacimiento.anyo = 1999;
300   pedro.fechaNacimiento.mes = 8;
301   pedro.fechaNacimiento.dia = 30;
302   pedro.edad =
          calcularEdad(hoy.diferenciaFechas(pedro.fechaNacimiento));
303   pedro.sexo = false;
304   pedro.email = "petermese@um.es";
305   pedro.notaMedia = 0.0;

307   pedro.asignaturas = new Asignatura[3];
308   pedro.asignaturas[0] = new Asignatura();
309   pedro.asignaturas[0].nombre = "Programación";
310   pedro.asignaturas[0].creditos = 6;
311   pedro.asignaturas[0].cuatrimestre = 2;
312   pedro.asignaturas[0].calificacion = random(0, 10);

314   pedro.asignaturas[1] = new Asignatura();
315   pedro.asignaturas[1].nombre = "Álgebra";
316   pedro.asignaturas[1].creditos = 6;
317   pedro.asignaturas[1].cuatrimestre = 1;
318   pedro.asignaturas[1].calificacion = random(0, 10);

320   pedro.asignaturas[2] = new Asignatura();
321   pedro.asignaturas[2].nombre = "Matemática discreta";
322   pedro.asignaturas[2].creditos = 6;
323   pedro.asignaturas[2].cuatrimestre = 2;
324   pedro.asignaturas[2].calificacion = random(0, 10);

326   marina.nombre = "Marina";
327   marina.apellidos = "Antelo Gutiérrez";
328   marina.fechaNacimiento = new Fecha();
329   marina.fechaNacimiento.anyo = 2003;
330   marina.fechaNacimiento.mes = 1;
331   marina.fechaNacimiento.dia = 22;
332   marina.edad =
          calcularEdad(hoy.diferenciaFechas(marina.fechaNacimiento));
333   marina.sexo = true;
334   marina.email = "maguanto@um.es";
335   marina.notaMedia = 0.0;
```

```
337   marina.asignaturas = new Asignatura[3];
338   marina.asignaturas[0] = new Asignatura();
339   marina.asignaturas[0].nombre = "Programación";
340   marina.asignaturas[0].creditos = 6;
341   marina.asignaturas[0].cuatrimestre = 2;
342   marina.asignaturas[0].calificacion = random(0, 10);

344   marina.asignaturas[1] = new Asignatura();
345   marina.asignaturas[1].nombre = "Álgebra";
346   marina.asignaturas[1].creditos = 6;
347   marina.asignaturas[1].cuatrimestre = 1;
348   marina.asignaturas[1].calificacion = random(0, 10);

350   marina.asignaturas[2] = new Asignatura();
351   marina.asignaturas[2].nombre = "Matemática discreta";
352   marina.asignaturas[2].creditos = 6;
353   marina.asignaturas[2].cuatrimestre = 2;
354   marina.asignaturas[2].calificacion = random(0, 10);

356   for (int i = 0; i < clase.alumnos.length; i++) {
357     Alumno alumno = clase.alumnos[i];
358     print("El alumno/a "+alumno.nombre+" "
359       +alumno.apellidos+" nació el "
360       +alumno.fechaNacimiento.dia+"/"
361       +alumno.fechaNacimiento.mes+"/"
362       +alumno.fechaNacimiento.anyo
363       +" (por lo tanto tiene "
364       +alumno.edad.anyos+" años, "
365       +alumno.edad.meses+" meses y "
366       +alumno.edad.dias+" días), su correo electrónico es "
            +alumno.email);
367     if (!alumno.sexo) {
368       println(" y es un hombre");
369     } else {
370       println(" y es una mujer");
371     }

373     for (int j = 0; j < alumno.asignaturas.length; j++) {
374       println("En la asignatura "+alumno.asignaturas[j].nombre+" ("
375         +alumno.asignaturas[j].creditos+" créditos, cuatrimestre "
376         +alumno.asignaturas[j].cuatrimestre+") "+alumno.nombre
377         +" tiene un "+alumno.asignaturas[j].calificacion);
378       alumno.notaMedia += alumno.asignaturas[j].calificacion;
379     }
380     alumno.notaMedia = alumno.notaMedia / alumno.asignaturas.length;
381     println("La nota media de todas las asignaturas de "+alumno.nombre
382       +" es: "+alumno.notaMedia);
383     println();
384   }

386   clase.ordenarAlumnosNotaMedia(true);
387   println("Si ordenamos los alumnos por su nota media, tenemos:");
388   for (int i = 0; i < clase.alumnos.length; i++) {
389     Alumno alumno = clase.alumnos[i];
390     println(alumno.nombre+" tiene un "+alumno.notaMedia);
```

```
391     }
392     println();

394     Alumno mejorAlumno = clase.mejorAlumnoClase();
395     println("Así las cosas, el mejor alumno de clase es "
396     +mejorAlumno.nombre+" con una nota media de "
            +mejorAlumno.notaMedia+"\n");

398     String[] nombresAsignaturas = {"Álgebra", "Matemática discreta", "
            Programación"};
399     for (int i = 0; i < nombresAsignaturas.length; i++) {
400        mejorAlumno = clase.mejorAlumnoAsignatura(nombresAsignaturas[i]);
401        println("El mejor alumno de "+nombresAsignaturas[i]+" es "
               +mejorAlumno.nombre);

403        clase.ordenarAlumnosPorAsignatura(true, nombresAsignaturas[i]);
404        float[] notas = clase.notasAsignatura(nombresAsignaturas[i]);
405        print("Todas las notas de "+nombresAsignaturas[i]+" son: [");
406        for (int j = 0; j < notas.length-1; j++) {
407          print(notas[j]+", ");
408        }
409        println(notas[notas.length-1]+"]");
410     }
411     println();

413     Edad mediaEdad = clase.mediaEdadClase();
414     println("Por último, la media de edad de la clase son "
415     +mediaEdad.anyos+" años, "+mediaEdad.meses+" meses y "
416     +mediaEdad.dias+" días");
417 }
```

Parte III

Ejemplos de exámenes



Examen 1

Ejercicio 1 (4.5 puntos) Dada una colección de matrices bidimensionales $(M_1, M_2, \ldots, M_n)$, representadas como una matriz tridimensional $\mathscr{M}$, escribe el código necesario para llevar a cabo las siguientes operaciones:

- Para cada matriz bidimensional M_i
  - **1.1** (1 punto) Si M_i contiene algún número primo, entonces calcular la multiplicación de todos los números primos que aparezcan en M_i.
  - **1.2** (1.5 puntos) En caso contrario, con un $p\,\%$ de probabilidad calcular la media aritmética de todos los elementos de M_i y con $(100-p)\,\%$ de probabilidad obtener un número aleatorio entre el elemento de menor valor y el elemento de mayor valor de M_i.

1.3 (1 punto) En función de un parámetro op dado, que determina la operación a realizar de entre las tres siguientes, calcular:

1. El máximo de todos los valores obtenidos para cada matriz M_i.
2. El mínimo de todos los valores obtenidos para cada matriz M_i.
3. La media aritmética de todos los valores obtenidos para cada matriz M_i.

1.4 (1 punto) Escribe un programa principal que:

- Solicite al usuario o genere aleatoriamente la matriz tridimensional $\mathscr{M}$ e imprimirla por pantalla.
- Solicite la probabilidad p.
- En función de todos estos valores $\mathscr{M}$ y p, ejecutar la operación correspondiente y mostrar por pantalla el resultado.

Nota:

Definir todas las funciones necesarias para que el código sea modular, extensible y reutilizable. Si algún apartado no se sabe hacer, se puede especificar la cabecera de la función y seguir por el siguiente apartado.

Ejercicio 2 (1.5 puntos) Escribe un programa que incluya una función que dada una cadena (String) C_1 devuelva el número de veces que aparece la subcadena C_2 en dicha cadena C_1, sin distinguir entre mayúsculas y minúsculas.

Por ejemplo, Si C_1 =“abbaabba” y C_2 =“ab”, entonces la función debería devolver un 2.
Nota de este ejercicio:

- No es necesario considerar vocales con acento.
- Se podrá hacer uso de funciones como `charAt()` y `length()`, pero no otras de la clase String.

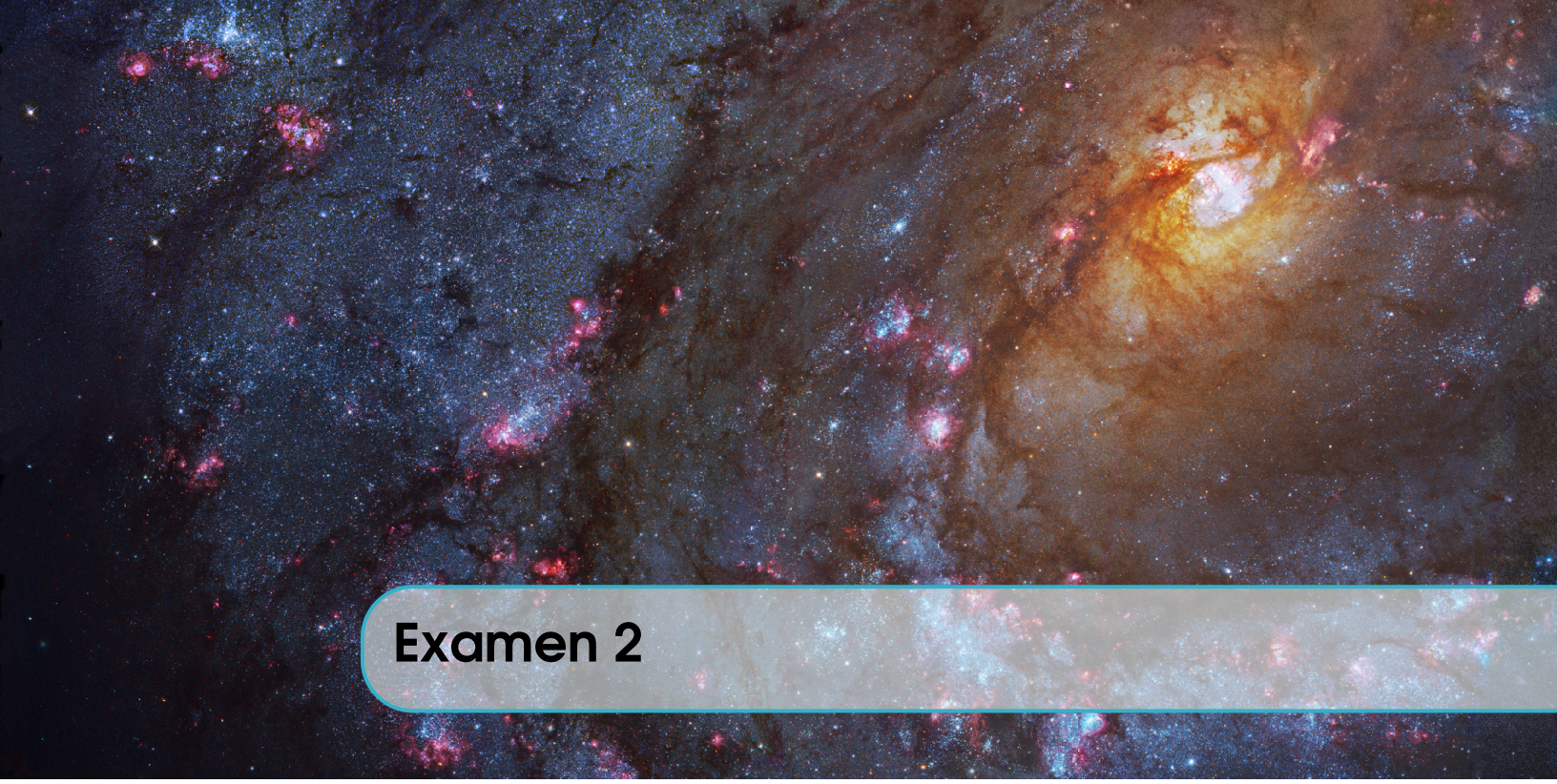
Ejercicio 3 (1.5 puntos) Escribe un programa que dado un array que contiene en cada una de sus posiciones las cifras de un número natural determine de manera recursiva la suma de dichas cifras.

Ejercicio 4 (2.5 puntos) Escribe un programa que incluya lo siguiente:

4.1 (0.5 puntos) Una clase para representar puntos en el espacio de coordenadas cartesianas.

4.2 (0.5 puntos) Una clase para representar triángulos definidos por sus tres vértices y su color. Permitir calcular el área y el perímetro de cualquier objeto perteneciente a la clase. Se debe hacer uso de constructores y del operador `this`.

4.3 (1.5 puntos) Una Pila es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se apilan y desapilan elementos. Los elementos se apilan uno detrás de otro y, a la hora de desapilar, se elimina el último elemento que haya sido apilado.
Construye una clase que represente una Pila sin tamaño máximo de objetos de la clase Triángulo, con las operaciones para apilar y desapilar.



Examen 2

Ejercicio 1 (1 punto) Reescribe el siguiente código usando únicamente bucles `while()` y describe su comportamiento a través de una fórmula matemática:

```
int f(int b) {
  int d = 1;
  for (int a = 1; a < b; a++) {
    int e = 0;
    for (int c = a; c >= 0; c--) {
      e += c;
    }
    d *= e;
  }
  return d;
}
```

Ejercicio 2 (2.5 puntos) Haz un programa que incluya lo siguiente:

2.1 (1 punto) Un número abundante o número excesivo es un número para el cual la suma de sus divisores propios es mayor que el propio número. Por otra parte, un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos. Escribe una función para determinar si un número dado es abundante, y otra función para determinar si un número dado es perfecto.

2.2 (1 punto) Obtener todos los números abundantes o perfectos, pero distintos de un número dado n, que haya en un intervalo $[a,b]$.

2.3 (0.5 puntos Buscar todos los números perfectos inferiores o iguales a un número dado.

Ejercicio 3 (4 puntos) Haz un programa que:

3.1 (1.5 puntos) Dado un array de n elementos, le asigne valores aleatorios en el intervalo $[min, max)$ hasta que la suma de todos los elementos sea mayor o igual a un valor v dado. Si se alcanza el final de array, pero la suma aún no cumple el criterio establecido, se debe continuar de nuevo por el primer elemento del array, de forma circular, reemplazando los valores existentes. El código debe evitar de antemano la ejecución de un bucle infinito, cuando sea imposible alcanzar la suma indicada.

3.2 (1,5 puntos) Dada una lista de n números naturales, comprobar si contiene los números del 1 al n, sin repetición y sin necesidad de que estén ordenados.

3.3 (1 punto) Dados dos conjuntos de enteros, obtener la unión de ambos, permitiendo elementos repetidos.

Ejercicio 4 (1 punto) Haz un programa que implemente una función recursiva para calcular el máximo común divisor de dos números a y b.

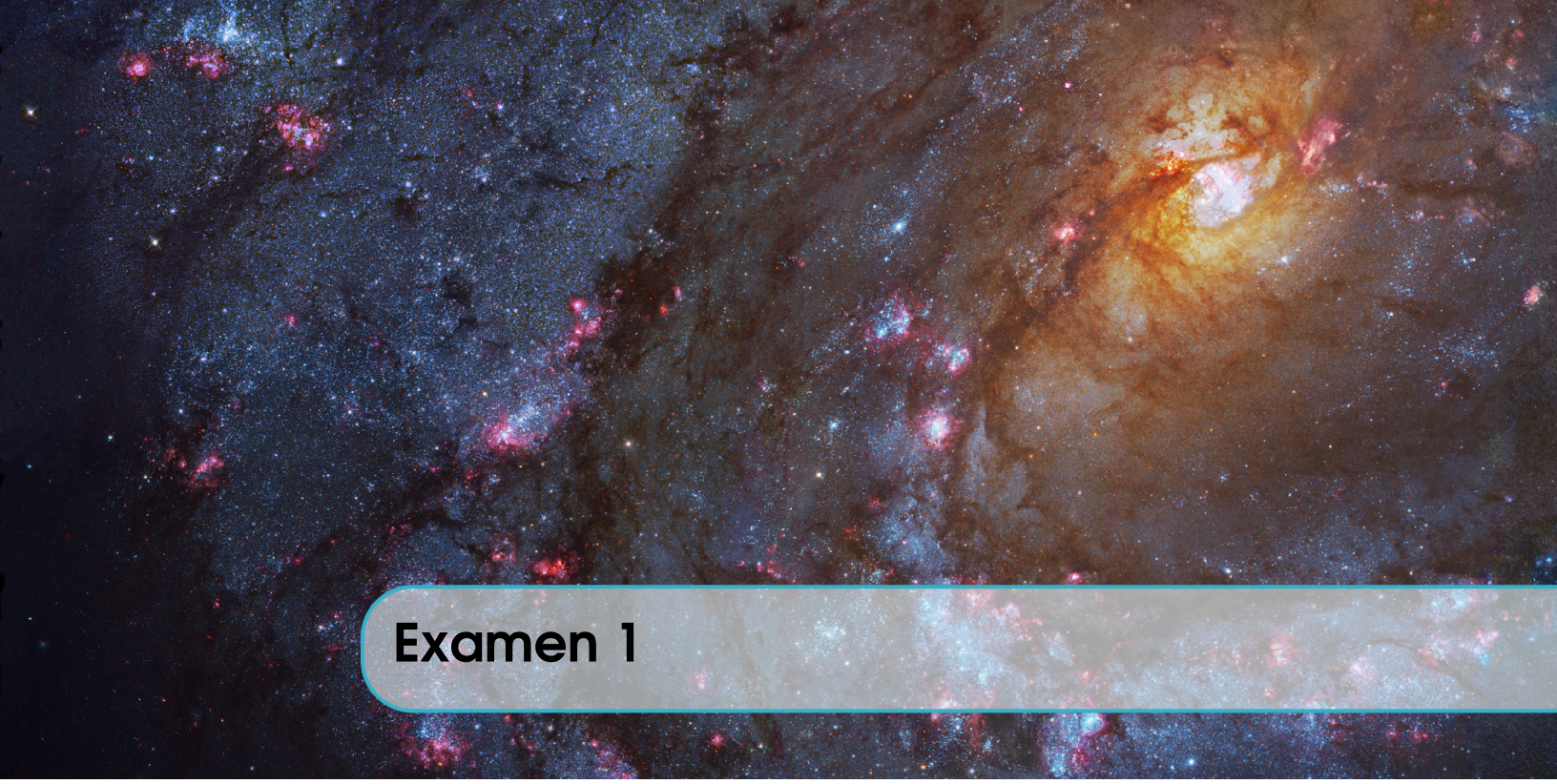
Ejercicio 5 (1,5 puntos) Escribe un programa que incluya lo siguiente:

5.1 (0.5 puntos) Una clase para representar un cilindro definido por su base y su altura, sobre el que se pueda calcular su superficie.

5.2 (1 punto) Dada una lista de cilindros, ordenar dicha lista en orden ascendente o descendente, según se indique, usando como criterio de ordenación la superficie del cilindro.

Parte IV

Ejemplos de exámenes y soluciones



Examen 1

Ejercicio 1 (4.5 puntos) Dada una colección de matrices bidimensionales $(M_1, M_2, \ldots, M_n)$, representadas como una matriz tridimensional $\mathscr{M}$, escribe el código necesario para llevar a cabo las siguientes operaciones:

- Para cada matriz bidimensional M_i
  - **1.1** (1 punto) Si M_i contiene algún número primo, entonces calcular la multiplicación de todos los números primos que aparezcan en M_i.

```
1  boolean esPrimo(int n) {
2    for (int i = 2; i < n; i++) {
3      if (n%i == 0) {
4        return false;
5      }
6    }
7    return true;
8  }

10 boolean contienePrimos(int[][] matriz){
11   for (int i = 0; i < matriz.length; i++) {
12     for (int j = 0; j < matriz[0].length; j++) {
13       if (esPrimo(matriz[i][j])) {
14         return true;
15       }
16     }
17   }

19   return false;
20 }

22 int multiplicarPrimos(int[][] matriz){
23   int resultado = 1;

25   for (int i = 0; i < matriz.length; i++) {
26     for (int j = 0; j < matriz[0].length; j++) {
```

```
27        if (esPrimo(matriz[i][j])) {
28          resultado = resultado * matriz[i][j];
29        }
30      }
31    }

33    return resultado;
34  }
```

1.2 (1.5 puntos) En caso contrario, con un $p\,\%$ de probabilidad calcular la media aritmética de todos los elementos de M_i y con $(100-p)\,\%$ de probabilidad obtener un número aleatorio entre el elemento de menor valor y el elemento de mayor valor de M_i.

```
1   float calcularMedia(int[][] matriz) {
2     if ((matriz == null) || (matriz.length < 1) ||
          (matriz[0].length < 1)) {
3       return 0.0;
4     }

6     float media = 0.0;

8     for (int i = 0; i < matriz.length; i++) {
9       for (int j = 0; j < matriz[0].length; j++) {
10        media += matriz[i][j];
11      }
12    }

14    return media/(matriz.length * matriz[0].length);
15  }

17  int minMatriz(int[][] matriz) {
18    if ((matriz == null) || (matriz.length < 1) ||
          (matriz[0].length < 1)) {
19      return 0;
20    }

22    int minimo = matriz[0][0];
23    for (int i = 0; i < matriz.length; i++) {
24      for (int j = 0; j < matriz[0].length; j++) {
25        if (matriz[i][j] < minimo) {
26          minimo = matriz[i][j];
27        }
28      }
29    }

31    return minimo;
32  }

34  int maxMatriz(int[][] matriz) {
35    if ((matriz == null) || (matriz.length < 1) ||
          (matriz[0].length < 1)) {
36      return 0;
37    }

39    int maximo = matriz[0][0];
40    for (int i = 0; i < matriz.length; i++) {
```

```
41      for (int j = 0; j < matriz[0].length; j++) {
42        if (matriz[i][j] > maximo) {
43          maximo = matriz[i][j];
44        }
45      }
46    }

48    return maximo;
49  }
```

1.3 (1 punto) En función de un parámetro op dado, que determina la operación a realizar de entre las tres siguientes, calcular:

1. El máximo de todos los valores obtenidos para cada matriz M_i.
2. El mínimo de todos los valores obtenidos para cada matriz M_i.
3. La media aritmética de todos los valores obtenidos para cada matriz M_i.

```
1  final int MIN = 0;
2  final int MAX = 1;
3  final int AVG = 2;

5  float operacionEnMatriz(int[][] matriz, int op) {
6    switch(op) {
7      case MIN:
8        return (float)minMatriz(matriz);
9      case MAX:
10       return (float)maxMatriz(matriz);
11     case AVG:
12       return calcularMedia(matriz);
13   }
14   return 0.0;
15 }
```

1.4 (1 punto) Escribe un programa principal que:

- Solicite al usuario o genere aleatoriamente la matriz tridimensional $\mathscr{M}$ e imprimirla por pantalla.
- Solicite la probabilidad p.
- En función de todos estos valores $\mathscr{M}$ y p, ejecutar la operación correspondiente y mostrar por pantalla el resultado.

```
1  import javax.swing.JOptionPane;

3  //Ejercicio 1.4
4  /*
5   * Crea una colección de matrices de un rango dado con valores
6   * aleatorios entre un mínimo y un máximo
7   */
8  int[][][] crearMatriz3DAleatoria(int matrices, int filas, int
       columnas, int min, int max) {
9    if ((matrices < 1) || (filas < 1) || (columnas < 1) || (max < min))
         {
10     return null;
11   }

13   int[][][] matriz3D = new int[matrices][filas][columnas];
```

```
15    for (int i = 0; i < matrices; i++) {
16      for (int j = 0; j < filas; j++) {
17        for (int k = 0; k < columnas; k++) {
18          matriz3D[i][j][k] = (int)random(min, max+1);
19        }
20      }
21    }

23    return matriz3D;
24  }

26  /*
27   * Genera y devuelve una representación de tipo
28   * String de una matriz dada.
29   */
30  String matrizAString(int[][][] matriz){
31    String cadena = "";

33    if (matriz == null){
34      return cadena;
35    }

37    for (int i = 0; i < matriz.length; i++) {
38      cadena += "A["+i+"] = \n";
39      for (int j = 0; j < matriz[0].length; j++) {
40        for (int k = 0; k < matriz[0][0].length; k++) {
41          cadena += matriz[i][j][k] + "\t";
42        }
43        cadena += "\n";
44      }
45      cadena += "\n\n";
46    }
47    return cadena;
48  }

50  /*
51   * Muestra por pantalla los valores de una
52   * matriz dada
53   */
54  void mostrarMatriz(int[][][] matriz){
55     print(matrizAString(matriz));
56  }

58  void setup(){
59    int matrices = 3;
60    int filas = 4;
61    int columnas = 5;
62    int min = 5;
63    int max = 15;
64    float p = float(JOptionPane.showInputDialog("Introduce una
          probabilidad entre 0 y 100: "));

66    int[][][] miMatriz = crearMatriz3DAleatoria(matrices, filas,
          columnas, min, max);
67    println("Mis "+matrices+" matrices aleatorias con "+filas+" filas y
          "+columnas+" columnas, con valores entre "+min+" y "+max+"
```

```
      son:");
68    mostrarMatriz(miMatriz);
69    println();

71    //Ejercicio 1.1
72    for (int i = 0; i < miMatriz.length; i++) {
73      if (contienePrimos(miMatriz[i])) {
74        int resultado = multiplicarPrimos(miMatriz[i]);
75        println("La matriz A["+i+"] sí contiene números primos y su
              multiplicación es: "+resultado);
76      } else {
77        print("La matriz A["+i+"] no contiene números primos ");
78        if (p < random(100)) {
79          float media = calcularMedia(miMatriz[i]);
80          println("y la media de sus valores es: "+media);
81        } else {
82          int minimo = minMatriz(miMatriz[i]);
83          int maximo = maxMatriz(miMatriz[i]);
84          int valor = (int)random(min, max+1);
85          println("y el valor aleatorio entre "+minimo+" y "+maximo+"
                es: "+valor);
86        }
87      }
88    }
89  }
```

Nota:

Definir todas las funciones necesarias para que el código sea modular, extensible y reutilizable. Si algún apartado no se sabe hacer, se puede especificar la cabecera de la función y seguir por el siguiente apartado.

Ejercicio 2 (1.5 puntos) Escribe un programa que incluya una función que dada una cadena (String) C_1 devuelva el número de veces que aparece la subcadena C_2 en dicha cadena C_1, sin distinguir entre mayúsculas y minúsculas.
Por ejemplo, Si C_1 =“abbaabba” y C_2 =“ab”, entonces la función debería devolver un 2.
Nota de este ejercicio:

- No es necesario considerar vocales con acento.
- Se podrá hacer uso de funciones como `charAt()` y `length()`, pero no otras de la clase String.

```
1  int contarSubcadena(String cadena1, String cadena2) {
2    int contador = 0;
3    int longitudCadena1 = cadena1.length();
4    int longitudCadena2 = cadena2.length();

6    // Iterar sobre cada caracter de la cadena principal
7    for (int i = 0; i <= longitudCadena1 - longitudCadena2; i++) {
8      // Inicializar una bandera para indicar si la subcadena es igual a la cadena objetivo
9      boolean esIgual = true;

11     // Comparar cada caracter de la subcadena con la cadena objetivo
12     for (int j = 0; j < longitudCadena2; j++) {
13       if (cadena1.charAt(i + j) != cadena2.charAt(j)) {
14         esIgual = false;
15         break;
```

```
      }
    }

    // Incrementar el contador si la subcadena es igual a la cadena objetivo
    if (esIgual) {
      contador++;
    }
  }

  return contador;
}

void setup() {
  // Ejemplo de uso
  String cadena1 = "Hola hola hola HOLA";
  String cadena2 = "hola";

  int resultado = contarSubcadena(cadena1, cadena2);

  println("La subcadena '" + cadena2 + "' aparece " + resultado + " veces
      en la cadena '" + cadena1 + "'");
}
```

Ejercicio 3 (1.5 puntos) Escribe un programa que dado un array que contiene en cada una de sus posiciones las cifras de un número natural determine de manera recursiva la suma de dichas cifras.

```
int sumaCifras(int[] numero, int indice) {
  // Caso base: cuando el índice es igual a la longitud del array
  if (indice == numero.length) {
    return 0;
  } else {
    // Llamada recursiva para sumar las cifras restantes
    int sumaRestante = sumaCifras(numero, indice + 1);

    // Sumar la cifra actual al resultado de la llamada recursiva
    return numero[indice] + sumaRestante;
  }
}

void setup() {
  int[] numero = {1, 2, 3, 4, 5};

  int suma = sumaCifras(numero, 0);

  println("La suma de las cifras es: " + suma);
}
```

Ejercicio 4 (2.5 puntos) Escribe un programa que incluya lo siguiente:

4.1 (0.5 puntos) Una clase para representar puntos en el espacio de coordenadas cartesianas.

```
class Punto {
  float coordX;
  float coordY;

  Punto (float coordX, float coordY) {
    this.coordX = coordX;
```

```
    this.coordY = coordY;
  }

  float distanciaEuclidea(Punto punto) {
    return sqrt(pow(coordX - punto.coordX, 2) + pow(coordY -
        punto.coordY, 2));
  }

  String toString() {
    return "("+coordX+", "+coordY+")";
  }
}

void setup() {
  int N = 10;
  float min = 1.0;
  float max = 10.0;

  for (int i = 0; i < N; i++) {
    Punto punto1 = new Punto(random(min, max), random(min, max));
    Punto punto2 = new Punto(random(min, max), random(min, max));

    float distancia = punto1.distanciaEuclidea(punto2);

    println("La distancia euclídea del punto "+punto1.toString()+" al
        punto "+punto2.toString()+" es "+distancia+"\n");
  }
}
```

4.2 (0.5 puntos) Una clase para representar triángulos definidos por sus tres vértices y su color. Permitir calcular el área y el perímetro de cualquier objeto perteneciente a la clase. Se debe hacer uso de constructores y del operador `this`.

```
class Triangulo {
  Punto punto1;
  Punto punto2;
  Punto punto3;
  String colorTriangulo;

  Triangulo(Punto punto1, Punto punto2, Punto punto3, String
      colorTriangulo) {
    this.punto1 = punto1;
    this.punto2 = punto2;
    this.punto3 = punto3;
    this.colorTriangulo = colorTriangulo;
  }

  double calcularPerimetro() {
    double lado1 = punto1.distanciaEuclidea(punto2);
    double lado2 = punto2.distanciaEuclidea(punto3);
    double lado3 = punto3.distanciaEuclidea(punto1);

    return lado1 + lado2 + lado3;
  }

  double calcularArea() {
```

```
23      double lado1 = punto1.distanciaEuclidea(punto2);
24      double lado2 = punto2.distanciaEuclidea(punto3);
25      double lado3 = punto3.distanciaEuclidea(punto1);

27      double s = (lado1 + lado2 + lado3) / 2;

29      return Math.sqrt(s * (s - lado1) * (s - lado2) * (s - lado3));
30    }

32    String toString() {
33      return "("+punto1.toString()+", "+punto2.toString()+", "
            +punto3.toString()+")";
34    }
35  }

37  void setup() {
38    int N = 10;
39    float min = 1.0;
40    float max = 10.0;
41    String[] colores = {"Rojo", "Azul", "Amarillo", "Verde"};

43    for (int i = 0; i < N; i++) {
44      Punto punto1 = new Punto(random(min, max), random(min, max));
45      Punto punto2 = new Punto(random(min, max), random(min, max));
46      Punto punto3 = new Punto(random(min, max), random(min, max));
47      String colorTriangulo = colores[(int)random(colores.length)];

49      Triangulo miTriangulo = new Triangulo(punto1, punto2, punto3,
            colorTriangulo);

51      double perimetro = miTriangulo.calcularPerimetro();
52      double area = miTriangulo.calcularArea();

54      println("Triángulo "+miTriangulo.toString()+" de color " +
            miTriangulo.colorTriangulo);
55      println("Perímetro: " + perimetro);
56      println("Área: " + area+"\n");
57    }
58  }
```

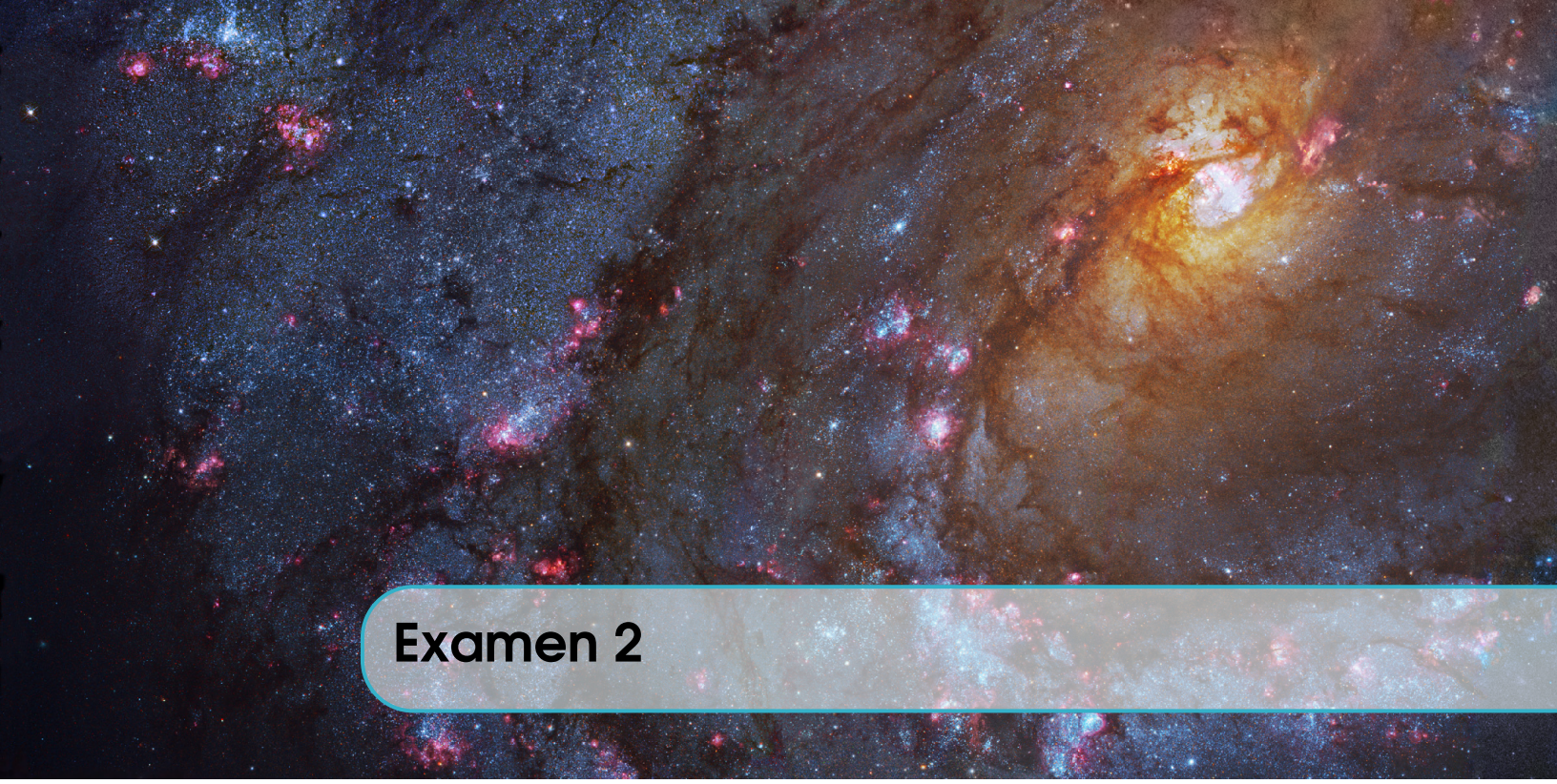
4.3 (1.5 puntos) Una Pila es una estructura de datos que puede usarse como un tipo de datos abstracto (TDA), y que funciona como una colección en la que se apilan y desapilan elementos. Los elementos se apilan uno detrás de otro y, a la hora de desapilar, se elimina el último elemento que haya sido apilado.

Construye una clase que represente una Pila sin tamaño máximo de objetos de la clase Triángulo, con las operaciones para apilar y desapilar.

```
1   class Pila {
2     Triangulo[] contenido;
3     int numElementos = 0;

5     void apilar(Triangulo triangulo) {
6       if ((contenido != null) && (numElementos < contenido.length)) {
7         contenido[numElementos] = triangulo;
8         numElementos++;
9       }
```

```
10    }

12    Triangulo desapilar(){
13      if (numElementos > 0) {
14        numElementos--;
15        return contenido[numElementos];
16      } else {
17        return null;
18      }
19    }

21    boolean estaVacia() {
22      return (numElementos == 0);
23    }

25    String toString() {
26      String s = "[";
27      for (int i = 0; i < numElementos-1; i++) {
28        s += contenido[i].toString()+"\n";
29      }
30      if (estaVacia()) {
31        return s += "]";
32      } else {
33        return s += contenido[numElementos-1].toString()+"]";
34      }
35    }
36  }


39  void setup() {
40    int N = 1;
41    float min = 1.0;
42    float max = 10.0;
43    String[] colores = {"Rojo", "Azul", "Amarillo", "Verde"};

45    float minCapacidad = 5;
46    float maxCapacidad = 10;

48    for (int i = 0; i < N; i++) {
49      Pila pila = new Pila();
50      int capacidad = (int)random(minCapacidad, maxCapacidad);
51      pila.contenido = new Triangulo[capacidad];

53      for (int j = 0; j < capacidad; j++) {
54        Punto punto1 = new Punto(random(min, max), random(min, max));
55        Punto punto2 = new Punto(random(min, max), random(min, max));
56        Punto punto3 = new Punto(random(min, max), random(min, max));
57        String colorTriangulo = colores[(int)random(colores.length)];

59        Triangulo miTriangulo = new Triangulo(punto1, punto2, punto3,
              colorTriangulo);
60        pila.apilar(miTriangulo);
61        println(pila.toString()+"\n");
62      }
63      println();
```

```
65      while(!pila.estaVacia()) {
66        Triangulo miTriangulo = pila.desapilar();
67        println(pila.toString()+"\n");
68      }
69    }
70  }
```



Examen 2

Ejercicio 1 (1 punto) Reescribe el siguiente código usando únicamente bucles `while()` y describe su comportamiento a través de una fórmula matemática:

```
int f(int b) {
  int d = 1;
  for (int a = 1; a < b; a++) {
    int e = 0;
    for (int c = a; c >= 0; c--) {
      e += c;
    }
    d *= e;
  }
  return d;
}
```

El código usando solamente bucles `while` es:

```
int f(int b) {
  int d = 1;
  int a = 1;
  while (a < b) {
    int e = 0;
    int c = a;
    while (c >= 0) {
      e += c;
      c--;
    }
    d *= e;
    a++;
  }
  return d;
}
```

Y la fórmula matemática que describe su comportamiento es:

$$f(b) = \prod_{a=1}^{b-1} \sum_{c=0}^{a} c$$

Ejercicio 2 (2.5 puntos) Haz un programa que incluya lo siguiente:

2.1 (1 punto) Un número abundante o número excesivo es un número para el cual la suma de sus divisores propios es mayor que el propio número. Por otra parte, un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos. Escribe una función para determinar si un número dado es abundante, y otra función para determinar si un número dado es perfecto.

```
boolean esAbundante(int num) {
  int sumaDivisores = 0;

  // Calcular la suma de los divisores propios del número
  for (int i = 1; i < num; i++) {
    if (num % i == 0) {
      sumaDivisores += i;
    }
  }

  // Verificar si la suma de los divisores es mayor que el número
  return (sumaDivisores > num);
}

boolean esPerfecto(int num) {
  int sumaDivisores = 0;

  // Calcular la suma de los divisores propios del número
  for (int i = 1; i < num; i++) {
    if (num % i == 0) {
      sumaDivisores += i;
    }
  }

  // Verificar si la suma de los divisores es igual al número
  return (sumaDivisores == num);
}
```

2.2 (1 punto) Obtener todos los números abundantes o perfectos, pero distintos de un número dado n, que haya en un intervalo $[a,b]$.

```
int[] numerosAbundantesPerfectosEnIntervalo(int a, int b, int n) {
  int cantidadNumeros = 0;
  for (int num = a; num <= b; num++) {
    if ((esAbundante(num) || esPerfecto(num)) && (num != n)) {
      cantidadNumeros++;
    }
  }

  int[] resultados = new int[cantidadNumeros];
  int indice = 0;
  for (int num = a; num <= b; num++) {
    if ((esAbundante(num) || esPerfecto(num)) && (num != n)) {
```

```
      resultados[indice] = num;
      indice++;
    }
  }

  return resultados;
}
```

2.3 (0.5 puntos Buscar todos los números perfectos inferiores o iguales a un número dado.

```
int[] numerosPerfectosHasta(int limite) {
  int cantidadNumeros = 0;
  for (int num = 1; num <= limite; num++) {
    if (esPerfecto(num)) {
      cantidadNumeros++;
    }
  }

  int[] perfectos = new int[cantidadNumeros];
  int indice = 0;
  for (int num = 1; num <= limite; num++) {
    if (esPerfecto(num)) {
      perfectos[indice] = num;
      indice++;
    }
  }

  return perfectos;
}
```

Y el programa principal para el ejercicio 2 es:

```
void setup() {
  int N = 10;
  int min = 0;
  int max = 100;
  // Ejercicio 2.1
  for (int i = 0; i < N; i++) {
    int numero = (int) random(min, max);
    if (esAbundante(numero)) {
      println("El número "+numero+" es abundante");
    } else {
      println("El número "+numero+" no es abundante");
    }

    if (esPerfecto(numero)) {
      println("El número "+numero+" es abundante");
    } else {
      println("El número "+numero+" no es abundante");
    }
  }

  // Ejercicio 2.2
  int n = (int)random(min, max);
  int[] numerosAbundantesPerfectos =
      numerosAbundantesPerfectosEnIntervalo(min, max, n);
  println("Números abundantes o perfectos en el intervalo ["+min+", "
      +max+"], pero distintos de "+n+": ");
```

```
  for (int i; i < numerosAbundantesPerfectos.length; i++) {
    println(numerosAbundantesPerfectos[i]);
  }

  // Ejercicio 2.3
  int[] numerosPerfectos = numerosPerfectosHasta(max);
  println("Números perfectos hasta "+max+": ");
  for (int i; i < numerosPerfectos.length; i++) {
    println(numerosPerfectos[i]);
  }
}
```

Ejercicio 3 (4 puntos) Haz un programa que:

3.1 (1.5 puntos) Dado un array de n elementos, le asigne valores aleatorios en el intervalo $[min, max)$ hasta que la suma de todos los elementos sea mayor o igual a un valor v dado. Si se alcanza el final de array, pero la suma aún no cumple el criterio establecido, se debe continuar de nuevo por el primer elemento del array, de forma circular, reemplazando los valores existentes. El código debe evitar de antemano la ejecución de un bucle infinito, cuando sea imposible alcanzar la suma indicada.

```
float[] generarArraySuma(float min, float max, float suma, int n) {
  // Evitamos ejecuciones infinitas del bucle
  if ((suma > (max*n)) || (suma < 0) || (min > max) || (max < 0)){
    return null;
  }

  // Crear el array de números (vacío)
  float[] array = new float[n];
  for (int i = 0; i < n; i++) {
    array[i] = 0;
  }

  // Recorrer el array en forma circular hasta alcanzar la suma objetivo
  int indice = 0;
  int sumaActual = 0;
  while (sumaActual < suma) {
    // Generar un nuevo valor aleatorio para el elemento actual
    array[indice] = random(min, max);

    // Pasar al siguiente índice en forma circular
    indice = (indice + 1) % n;

    // Actualizar la suma
    sumaActual = calcularSuma(array);
  }

  return array;
}


float calcularSuma(float[] array) {
  float sum = 0;
  for (int i = 0; i < array.length; i++) {
    sum += array[i];
  }
```

```
  return sum;
}
```

3.2 (1,5 puntos) Dada una lista de n números naturales, comprobar si contiene los números del 1 al n, sin repetición y sin necesidad de que estén ordenados.

```
boolean contieneNumerosNaturales(int[] lista) {
  int n = lista.length;

  // Crear un array auxiliar para marcar la presencia de los números
  boolean[] numerosPresentes = new boolean[n];
  for (int i = 0; i < numerosPresentes.length; i++) {
    numerosPresentes[i] = false;
  }

  // Marcar la presencia de los números en la lista
  for (int i = 0; i < lista.length; i++) {
  int numero = lista[i];
    if (numero < 1 || numero > n || numerosPresentes[numero-1]) {
      // Si el número está fuera del rango o ya está repetido, la lista no cumple
          la condición
      return false;
    }
    numerosPresentes[numero-1] = true;
  }

  // Si todos los números del 1 al n están presentes sin repetición, la lista
      cumple la condición
  return true;
}
```

3.3 (1 punto) Dados dos conjuntos de enteros, obtener la unión de ambos, permitiendo elementos repetidos.

```
int[] concatenarArrays(int[] array1, int[] array2) {
  // Crear un nuevo array con la longitud total de los dos arrays de entrada
  int[] resultado = new int[array1.length + array2.length];

  // Copiar los elementos del primer array al resultado
  for (int i = 0; i < array1.length; i++) {
    resultado[i] = array1[i];
  }

  // Copiar los elementos del segundo array al resultado
  for (int i = 0; i < array2.length; i++) {
    resultado[array1.length + i] = array2[i];
  }

  // Devolver el array resultado
  return resultado;
}
```

Y el programa principal para el ejercicio 3 es:

```
void setup() {
  // Ejercicio 3.1
  int n = 10; // Tamaño del array
  float min = 0.0; // Valor mínimo (inclusive)
```

```
5   float max = 1.0; // Valor máximo (exclusivo)
6   float suma = 7.0; // Suma objetivo

8   // Generar el array que cumple con la condición establecida
9   float[] array = generarArraySuma(min, max, suma, n);

11  // Imprimir el array resultante
12  print("Array resultante: ");
13  for (int i = 0; i < array.length; i++) {
14    print(value+" ");
15  }
16  println("\n");

18  // Ejercicio 3.2
19  int[] lista1 = {1, 2, 3, 4, 5};
20  int[] lista2 = {1, 3, 5, 2, 4};
21  int[] lista3 = {1, 2, 2, 3, 4};

23  print("La lista "+arrayAString(lista1));
24  if (contieneNumerosNaturales(lista1)) {
25    println("sí contiene todos los números sin repetición\n");
26  } else {
27    println("no contiene todos los números sin repetición\n");
28  }

30  print("La lista "+arrayAString(lista2));
31  if (contieneNumerosNaturales(lista2)) {
32    println("sí contiene todos los números sin repetición\n");
33  } else {
34    println("no contiene todos los números sin repetición\n");
35  }

37  print("La lista "+arrayAString(lista3));
38  if (contieneNumerosNaturales(lista3)) {
39    println("sí contiene todos los números sin repetición\n");
40  } else {
41    println("no contiene todos los números sin repetición\n");
42  }

44  // Ejercicio 3.3
45  int[] array1 = {1, 2, 3};
46  int[] array2 = {3, 4, 5, 6};

48  int[] resultado = concatenarArrays(array1, array2);

50  print("La unión de "+arrayAString(array1)+" y "
51                    +arrayAString(array1)+" es: "
52          +arrayAString(resultado));
53 }

55 String arrayAString(int[] array) {
56   String resultado = "";
57   for (int i = 0; i < array.length; i++) {
58     resultado += value+" ";
59   }
60   return resultado;
```

```
61 }
```

Ejercicio 4 (1 punto) Haz un programa que implemente una función recursiva para calcular el máximo común divisor de dos números a y b.

```
int calcularMCD(int a, int b) {
  if ((a < 0) || (b < 0)) {
  return -1;
  }

  // Caso base: si b es igual a 0, el MCD es a
  if (b == 0) {
    return a;
  }

  // Llamada recursiva: calcular el MCD de b y el resto de a dividido por b
  return calcularMCD(b, a % b);
}

void setup() {
  int a = 24;
  int b = 36;

  int mcd = calcularMCD(a, b);

  println("El máximo común divisor de " + a + " y " + b + " es: " + mcd);
}
```

Ejercicio 5 (1,5 puntos) Escribe un programa que incluya lo siguiente:

5.1 (0.5 puntos) Una clase para representar un cilindro definido por su base y su altura, sobre el que se pueda calcular su superficie.

```
class Círculo {
  float radio;

  float calcularArea() {
    return PI * radio * radio;
  }
}

class Cilindro {
  Círculo base;
  float altura;

  float calcularSuperficie() {
    float areaBase = base.calcularArea();
    float areaLateral = 2 * PI * base.radio * altura;
    return 2 * areaBase + areaLateral;
  }
}
```

5.2 (1 punto) Dada una lista de cilindros, ordenar dicha lista en orden ascendente o descendente, según se indique, usando como criterio de ordenación la superficie del cilindro.

```
void ordenarCilindros(Cilindro[] cilindros, boolean ascendente) {
  int n = cilindros.length;
```

```
  for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
      float superficie1 = cilindros[j].calcularSuperficie();
      float superficie2 = cilindros[j + 1].calcularSuperficie();
      boolean intercambiar;
      if (ascendente) {
        intercambiar = superficie1 > superficie2;
      } else {
        intercambiar = superficie1 < superficie2;
      }
      if (intercambiar) {
        Cilindro temp = cilindros[j];
        cilindros[j] = cilindros[j + 1];
        cilindros[j + 1] = temp;
      }
    }
  }
}
```

Y el programa principal para el ejercicio 5 es:

```
void setup() {
  // Ejercicio 5.1
  Círculo base = new Círculo();
  base.radio = 5;

  Cilindro cilindro = new Cilindro();
  cilindro.base = base;
  cilindro.altura = 10;

  float superficie = cilindro.calcularSuperficie();
  println("La superficie del cilindro es: " + superficie);

  // Ejercicio 5.2
  int N = 5;
  int min_radio = 2;
  int max_radio = 6;
  int min_altura = 8;
  int max_altura = 20;

  Cilindro cilindros = new Cilindro[N];
  for (int i = 0; i < cilindros.length; i++) {
      Círculo base = new Círculo();
    base.radio = (int)random(min_radio, max_radio);
    cilindros[i].base = base;
    cilindros[i].altura = (int)random(min_altura, max_altura);
  }

  // Ordenar el array de cilindros por superficie de forma ascendente
  ordenarCilindros(cilindros, true);

  // Imprimir el array ordenado de cilindros
  for (int i = 0; i < cilindros.length; i++) {
    float superficie = cilindros[i].calcularSuperficie();
    println("Superficie del cilindro: " + superficie);
  }
}
```