Definiciones:

- Cuadrante se define como un enumerado que puede tomar los valores: *PRIMER_CUADRANTE, SEGUNDO_CUADRANTE, TERCER_CUADRANTE, CUARTO_CUADRANTE*.
- El tipo LocalDate de Java 8 sirve para trabajar con fechas.
- Una tupla t de tamaño n es un agregado inmutable de elementos que pueden ser de diferentes tipos y lo escribimos como $s = (e_0, e_1, \dots, e_{n-1})$
- Una lista ls de tamaño n es un agregado indexable de elementos y lo escribimos como $ls = [e_0, e_1, \dots, e_{n-1}]$
- Una conjunto st de tamaño n es un agregado de elementos sin repetición que pueden ser de diferentes tipos y lo escribimos como $s = \{e_0, e_1, \dots, e_{n-1}\}$
- Una diccionario dt de tamaño n es un agregado de pares elementos claves-valor, sin repetición de las claves, y lo escribimos como $dt = \{k_0: v_0, k_1: v_1, \dots, k_{n-1}: v_{n-1}\}$. Es el tipo *Map<K, V>.*
- Multimap<K, V> es un tipo de la forma Map<K, List<V>>. Otra variante puede ser Map<K, Set<V>>.
- Multiset<V> es un tipo de la forma Map<V, Integer>

Ejercicios para resolver de forma iterativa con Java8, Java7 y posteriormente de forma recursiva. Un método en cada una de las formas para cada ejercicio. Decidir cual es la forma más adecuada para resolver cada problema.

1. Sumar de los elementos de una lista
2. Dada una lista de objetos de tipo Punto devolver otra con la coordenada X de esos puntos.
3. Dada una lista de enteros decidir si todos son impares.
4. Dada una lista de enteros decidir si existe alguno impar y primo.
5. Dada una lista de reales sumar de todos los elementos de la lista.
6. Dada una lista de enteros sumar los cuadrados de todos los elementos de la lista.
7. Dada una lista de reales buscar el primero que sea mayor que un umbral dado.
8. Dada una lista de Punto construir otra cuyos puntos sean simétricos con el respecto al eje Y.
9. Dada una lista de Punto buscar el de mayor coordenada X.
10. Dada una lista de Punto contar cuántos están en el primer cuadrante
11. Dado un array de puntos construir una lista con todos ellos.
12. Dada una *List<Punto>* agrupar los puntos por cuadrantes. Es decir, devolver un *Multimap* en el que se asocia a cada cuadrante los puntos del que están en él.
13. Dada una *List<Punto>* calcular la suma de las coordenadas X de los que están en cada cuadrante. Es decir, devuelva un *Map<Cuadrante, Double>* en el que se asocia a cada cuadrante la suma de las coordenadas X de los puntos de ese cuadrante.
14. Dada una *List<Punto>* construir un *Multiset* que guarde el número de puntos de cada cuadrante.

15. Dada una lista de enteros calcular el máximo, el mínimo, la media y la suma.
16. Decidir si un número es primo.
17. Buscar el primer primo mayor que un número dado
18. Contar el número de caracteres en minúscula que tiene una cadena.
19. Encontrar la inversa de una cadena de caracteres.
20. Dada una cadena de caracteres decidir si es un palíndromo. Una cadena es un palíndromo si es igual a su inversa.
21. Factorial de un número entero n
22. Máximo de los n elementos de un array de reales dt que contiene n elementos.
23. Suma de los dígitos del entero a en base r
24. Número de dígitos del entero a en base r
25. Calcular el máximo común divisor de dos enteros positivos $a,\ b$
26. Dados dos enteros positivos $a,\ b$ calcular su división entera y su resto usando la operación suma de los enteros.
27. Dados los enteros positivos $a,\ n$ calcular $m = \sqrt[n]{a}$. Siendo m el mayor entero que cumple $a \geq m^n$
28. Calcular a^n
29. Obtener la representación del entero e en base r
30. Buscar si todos los dígitos del entero e que cumplen el predicado $p(x)$
31. Sumar los dígitos de un entero e en base r
32. Calcular la suma del factorial de cada uno de los dígitos del número n en base r. Ejemplo: *sfactorial(1024,10) =1! + 0! + 2! + 4! = 1 +1 + 2 + 24 = 28*
33. Contar los dígitos de un entero e en base r. ¿Por qué es igual a $int(\log_r e) + 1$
34. Dado un entero n en base r obtener otro en la misma base cuyos dígitos estén invertidos. Ejemplo: sean inversos *invierte(1024,10)=4201*.
35. Dado un entero n obtener una lista con sus divisores.
36. Buscar si el elemento e está contenido en lista ls.
    a. Asumir que la lista no está ordenada
    b. Asumir que la lista está ordenada
37. Encontrar el índice del elemento e en lista ls y si no lo contiene entonces devolver -1.
    a. Asumir que la lista no está ordenada
    b. Asumir que la lista está ordenada
38. Buscar si existe un e contenido en lista ls que cumpla el predicado $p(x)$
39. Dada la lista ls y las String p, f, s construir un String que tenga p al principio, f al final y en medio los elementos de ls convertidos en String separados por s.
40. Obtener el producto escalar de dos vectores de tamaño n representados en forma de listas
41. Decidir si dos listas son iguales
42. Dadas dos listas de reales del mismo tamaño decidir si los incrementos de cada casilla con respecto a la anterior son mismo signo en ambas listas para todas las casillas.
43. Decidir si una lista está ordenada con respecto a un orden.
44. Dada lista de enteros decidir si el valor de alguna casilla es igual al valor del índice.

45. Decidir si los elementos de una lista de enteros forman una progresión aritmética
46. Dada una lista de elementos de tamaño n decidir si hay alguno de ellos que se repita $n/2$ veces o más.
47. Dada una lista de *String* buscar la cadena que tiene un mayor número de caracteres en minúscula.
48. Dada una *List<List<Integer>>* construir una *List<Integer>* que contenga los enteros de todas las listas dadas.
49. Un método que guarde en un fichero de texto los números primos hasta un número n dado.
50. Un método que guarde en un fichero de texto el cuadrado de los números primos hasta un número n dado.
51. Un método que devuelva un *List<Integer>* a partir de un fichero de texto que contiene en cada línea un número entero.
52. Dado un fichero de texto con una fecha escrita en cada línea, genere otro fichero con las fechas ordenadas y que estén entre dos fechas dadas.
53. Obtener una *List<Integer>* a partir de un fichero de texto que contiene en cada línea una lista de números enteros separados por comas.
54. Dadas dos listas ordenadas obtener otra cadena ordenada que contenga los elementos de las dos anteriores (fusión de listas)
55. Dada una lista y un elemento del mismo tipo que las casillas, que llamaremos pivote, reordenarla, de menor a mayor, para que resulten tres bloques: los menores que el pivote, los iguales al pivote y los mayores que el pivote. El algoritmo debe devolver dos enteros que son las posiciones de las casillas que separan los tres bloques formados (algoritmo de la bandera holandesa)